Generator Manager User Guide

Release 8.1.3

November 2013

IKAN Solutions N.V.
Schaliënhoevedreef 20A
B-2800 Mechelen
BELGIUM

Copyright © 2013, IKAN Solutions N.V.

No part of this document may be reproduced or transmitted in any form or by any means, electronically or mechanically, for any purpose, without the express written permission of IKAN Solutions N.V.

MetaSuite is a trademark of IKAN Solutions N.V.

Table of Contents

Chapter 1 - About This Manual 1

1.1. Related Publications 1

1.2. About MetaSuite 2

Chapter 2 - Logging on 3

Chapter 3 - Create Dictionary/Enter License Key Screen 6

3.1. Creating a Dictionary File 6

3.2. Entering a License Key 10

Chapter 4 - The Dictionary Options Screen 12

Chapter 5 - The Generate Screen - Overview 15

Chapter 6 - The Generate Screen - Implementing MIL Instructions 18

6.1. The Generate MIL Screen 19

6.2. Copying Existing MIL Files 20

6.3. Adding MIL Instructions Using the Command Wizard 21

6.4. CHANGE Options Reference 24

BUILD *25*

DATE FORMAT *25*

DATE BREAK YEAR *26*

DECIMAL *26*

EXEC *27*

INTERPUNCTION *27*

NULL *28*

NULLABLE *28*

PAGE *29*

QUOTE *29*

SQL *30*

SQL-QUOTE *30*

DYNAMIC *31*

LANGUAGE *31*

UNICODE-CCSID *31*

CHARACTER-CCSID *32*

6.5. TABLE Options Reference 32
LIST 33
ADD 33
COPY 34
DELETE 34
IMPORT 35
EXPORT 35

Chapter 7 - The Generate Screen - Working With MDL Files 36

7.1. The Generate MDL Screen 37
7.2. Editing Existing MDL files 38
7.3. Using MDL Tools - Make Delimited Target MDL 39
7.4. Using MDL Tools - Make XML Target MDL 41
7.5. Using MDL Tools - Parse XML Source 44

Chapter 8 - The Generate Screen - Working With MSM Files 46

Chapter 9 - The Generate Screen - Working With MXL Files 48

9.1. Generating an MXL file 49
9.2. Generating Multiple MXL Files in Batch 51

Chapter 10 - The Table Maintenance Screen - Overview 53

Chapter 11 - The Table Maintenance Screen - Managing MGL Tables 57

11.1. The MGL Table Maintenance Screen 58
11.2. MGL Tables: Detailed Description 63
11.3. Code-control Tables 64
Format 65
Source Code-control Tables 66
Target Code-control Tables 68
User-written Code-control Tables 71
11.4. COBOL Code Tables 71
11.5. Code Table Examples 72
Example 1 - Code Table for Calling I/O Subroutines 72
Example 2 - Code Table for Calling Reformatting Subroutines 73
11.6. Creating Customized Code-control Tables 73
11.7. Creating Customized COBOL Code Tables 74

Chapter 12 - The Table Maintenance Screen - Managing MRL Tables 75

12.1. The MRL Table Maintenance Screen 76

Chapter 13 - The Other Functions Screen 81

13.1. Generator Initial Settings 81
13.2. Browse Folders 82

Chapter 14 - Installation Language Commands 85

14.1. NEW 85
Columns 1-3 ..... *86*
nnnn *86*
14.2. CHANGE DEFAULT BUILD 86
Build Number *86*
Example *86*
14.3. CHANGE DEFAULT DATE 86
ISO *86*
EUR *87*
JIS *87*
USA *87*
(century,end-year) *87*
14.4. CHANGE DEFAULT DECIMAL 87
14.5. CHANGE DEFAULT EXEC 88
14.6. CHANGE DEFAULT INTERPUNCTION 88
OFF *88*
NO *88*
YES *88*
14.7. CHANGE DEFAULT NULL 89
Null_Character *89*
14.8. CHANGE DEFAULT NULLABLE 89
Nullable_Option *89*
14.9. CHANGE DEFAULT PAGE 89
14.10. CHANGE DEFAULT QUOTE 90
14.11. CHANGE DEFAULT SQL 90
14.12. CHANGE DEFAULT SQL-QUOTE 91
14.13. CHANGE DEFAULT DYNAMIC 91
14.14. CHANGE DEFAULT CHARACTER-CCSID 91
14.15. CHANGE DEFAULT UNICODE-CCSID 91
14.16. ADD TABLE 91
Table-name *92*
Entry-size *92*
Table-entry *92*
14.17. COPY TABLE 92
table-name *92*
14.18. DELETE TABLE 93
Table-name *93*
14.19. LIST DEFAULT 93
Example *93*

14.20. LIST TABLE 93
Table-name *93*
Example *94*
14.21. LIST VERSION 94
Example *94*
14.22. REMARKS 95

Chapter 15 - MetaSuite Template Language 96

15.1. MTL Variables, Expressions and Functions 96
MTL Variables *97*
Temporary MTL Variables *97*
MTL Expressions *98*
MTL Functions *99*
15.2. Generator Variables 101
15.3. MTL Commands 102
Notation conventions *103*
CALL table *104*
DUMP *105*
EVALUATE structures *105*
EXIT table *107*
Fault Message handling *107*
FOR-NEXT loops *110*
FREEZE and UNFREEZE *111*
GOTO *111*
IF structures *113*
IMPLEMENT *114*
NESTED IF *114*
REMARK *114*
SET *115*
SKIP *115*
TRACE *116*
UNSET *117*
15.4. MTL System Variables 118
15.5. SourceFile-specific MTL Variables 120
15.6. SourceRecord-related MTL Variables 121
15.7. TargetFile-related MTL Variables 122
15.8. TargetRecord-related Variables 123
15.9. Field-related MTL Variables 124

Chapter 16 - MetaSuite Run Language 127

16.1. Tables 127
16.2. Examples 130
Windows *130*
UNIX / Linux *131*
VMS *131*

OSD/BC BS2000 ..... *132*
OS/390 Z/OS *132*

Chapter 17 - String Modules 134

17.1. Description 134
17.2. Usage 135
17.3. INITSTR - Initialize parameter fields 135
Description *135*
Input *135*
Output *136*
17.4. LENSTR - Determine the length of a string 136
Description *136*
Input *137*
Output *137*
Remarks *137*
Examples *137*
17.5. SUBSTR - Return a string from a string 137
Description *138*
Input *138*
Output *138*
Remarks *138*
Examples *139*
17.6. SRCHSTR - Search in a string 139
Description *139*
Input *139*
Output *140*
Remarks *140*
Examples *140*
17.7. REPLSTR - Replace substring in a string 141
Description *141*
Input *141*
Output *142*
Remarks *142*
Examples *143*

Chapter 18 - Calling the Generator in Batch 144

18.1. Preparation 144
18.2. Using MSBGEN.EXE 144
18.3. Example 145

Chapter 19 - Generator Message Reference 147

19.1. Generator Return Codes 147
19.2. Generator Syntax Error Messages 150
19.3. Dictionary Error Messages 151

19.4. List of Generator Messages....151
Messages 001 to 050....*152*
Messages 051 to 100....*154*
Messages 101 to 150....*157*
Messages 151 to 200....*160*
Messages 201 to 250....*163*
Messages 251 to 300....*165*
Messages 301 to 350....*167*
Messages 351 to 400....*169*
Messages 401 to 450....*171*
Messages 451 to 500....*174*
Messages 501 to 550....*176*
Messages 551 to 600....*178*
Messages 601 to 650....*181*
Messages 651 to 700....*184*
Messages 701 to 711....*190*

CHAPTER 1
About This Manual

The MetaSuite Generator is a collection of COBOL building blocks, required to translate the MetaMap Model (MSM) into a COBOL application for the requested operating system platform.

The MetaSuite Generator Manager is a tool designed for helping MetaSuite Administrators to generate and manipulate the MetaSuite Generator Dictionary and the Dictionary Upload Files (DDL files).

The *Generator Manager Guide* is part of the MetaSuite documentation set intended for MetaSuite Administrators.

1.1. Related Publications

The following table gives an overview of the complete MetaSuite documentation set.

Release Information	Release Notes 8.1.3
Installation Guides	• BS2000/OSD Runtime Component • DOS/VSE Runtime Component • Fujitsu Windows Runtime Component • MicroFocus Windows Runtime Component • MicroFocus UNIX Runtime Component • OS/390 and Z/OS Runtime Component • OS/400 Runtime Component • VisualAge Windows Runtime Component • VisualAge UNIX Runtime Component • VMS Runtime Component
User Guides	• INI Manager User Guide • Installation and Setup Guide • Introduction Guide • MetaStore Manager User Guide • MetaMap Manager User Guide • Generator Manager User Guide
Technical Guides	• ADABAS File Access Guide • IDMS File Access Guide • IMS DLI File Access Guide • RDBMS File Access Guide • XML File Access Guide • Runtime Modules • User-defined Functions User Guide

1.2. About MetaSuite

If you are unfamiliar with MetaSuite, the following technical description provides you with a brief overview.

The MetaSuite System	MetaSuite is designed for data retrieval, extraction, conversion and reporting. It includes a workstation-based graphical user interface and a mainframe runtime component.
MetaSuite Database Interfaces	MetaSuite can access data from a number of database management systems, using the same commands, program structure and retrieval techniques used for non-database files. Each database interface is available as an optional enhancement to the base product.
MetaMap Manager	MetaMap Manager is the MetaSuite tool used to define models. Such models are intuitively built by describing overall program specifications, input file definitions (data and process) and target file definitions (data and process).
MetaStore Manager	MetaStore Manager is a tool that provides metadata maintenance and documentation services.
Generator Manager	The Generator Manager is the system administration tool. All kinds of basic functionalities and customization possibilities are supported by this tool.

CHAPTER 2
Logging on

1. Start the *MetaSuite Generator Manager*.

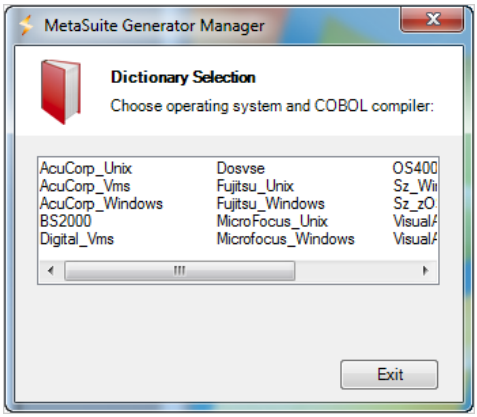
   Select MetaSuite Generator Manager from the MetaSuite shortcut menu or double-click the *GeneratorManager.exe* file in the MetaSuite installation folder.

2. If not found automatically, select the required INI file.

   If the default initialization file *MetaSuite.ini* is not found in the MetaSuite installation folder, the program will be (re)installed.

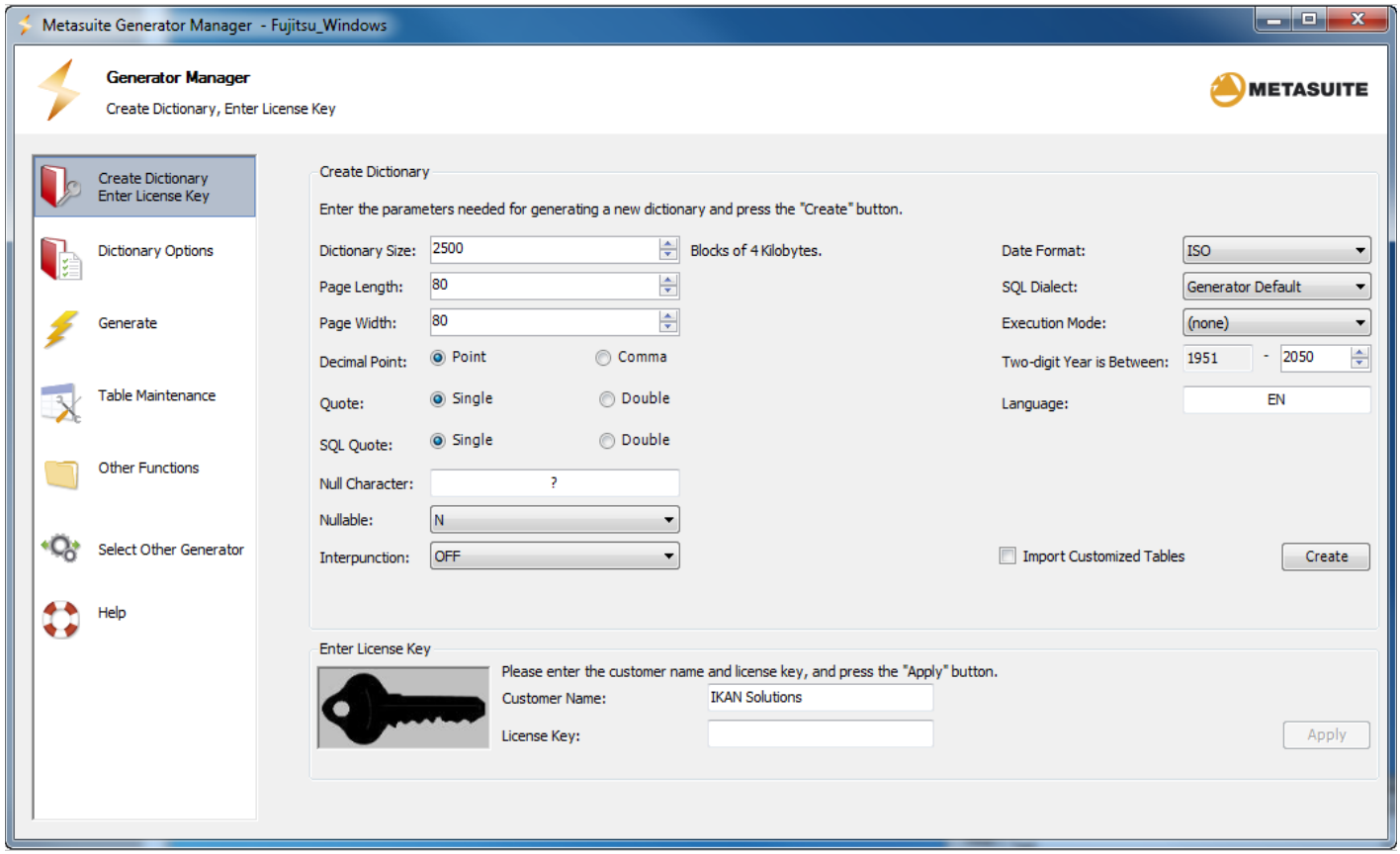
3. If no default Generator has been specified, you need to select the required MetaSuite Generator.

   A pop-up window listing the MetaSuite Generators installed in this configuration is displayed.



   Select the required Generator, or click *Exit* to leave the program.

4. The *MetaSuite Generator Manager* screen is displayed.



The icons in the left column of the window, give access to the different Generator Manager screens.

Icon	Generator Manager Screen
	Create Dictionary/Enter License Key Screen (page 6)
	The Dictionary Options Screen (page 12)
	The Generate Screen - Overview (page 15)
	The Table Maintenance Screen - Overview (page 53)
	The Other Functions Screen (page 81)

Icon	Generator Manager Screen
	Select Other Generator
	Help

CHAPTER 3

Create Dictionary/Enter License Key Screen

The *Create Dictionary, Enter License Key* screen is used to generate new or existing dictionary files, and to enter license keys.

Switch to this screen by clicking the *Create Dictionary/Enter License Key* icon in the left column.

Refer to the following procedures for a detailed description:

- Creating a Dictionary File (page 6)
- Entering a License Key (page 10)

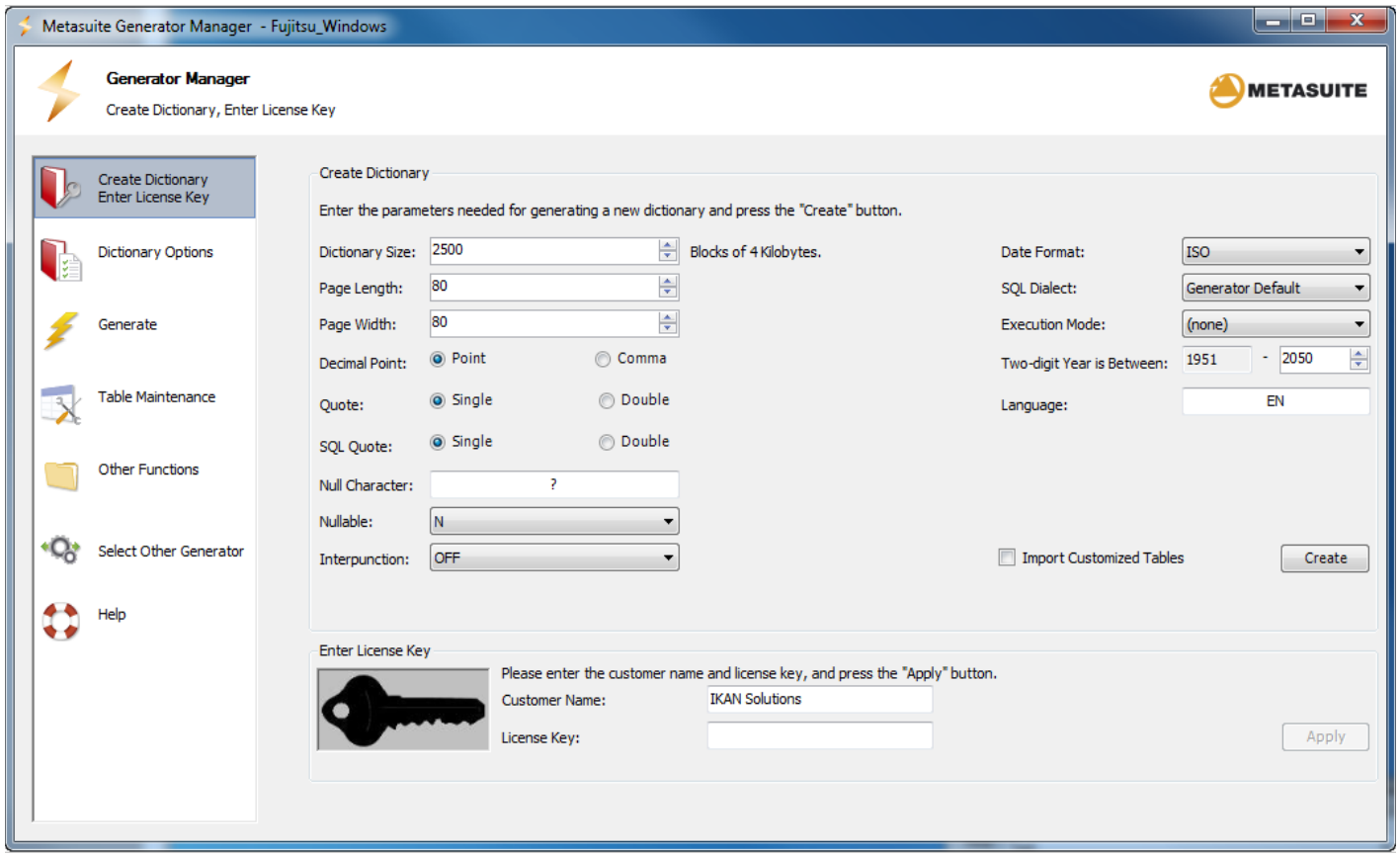
3.1. Creating a Dictionary File

Use this function to convert Dictionary Upload Files (DDL files) into a Dictionary which will be used during the generating process. Without such a Dictionary, no COBOL source can be generated. This operation is normally part of the initial setup process (See the *Installation and Setup Guide*).

Later on, you also need to generate a Dictionary File in order to take into account your changes to the MRL settings (page 75) and the MGL Tables (page 57).

1. Switch to the *Create Dictionary/Enter License Key* screen by clicking the appropriate icon in the left column.

   The following screen is displayed:



2. Fill out the *Create Dictionary* section.

   It contains the following fields:

Field	Description
Dictionary size	Enter the number of Generator library blocks to be initialized. You can enter a higher value (up to 9999) if your data transformations require the processing of very large Data Sources.
Page length	Enter the default page length (in number of lines) that is used for targets reports when no page settings are specified in the MetaMap model.
Page width	Enter the default page width (in number of characters) that is used for targets reports when no page settings are specified in the MetaMap model.
Decimal point	Select the required Decimal Point character. You can choose between *Point* and *Comma*.
Quote	Select the required Quote character. You can choose between *Single* and *Double*.
SQL Quote	Select the require Quote character for SQL commands. You can choose between *Single* and *Double*.

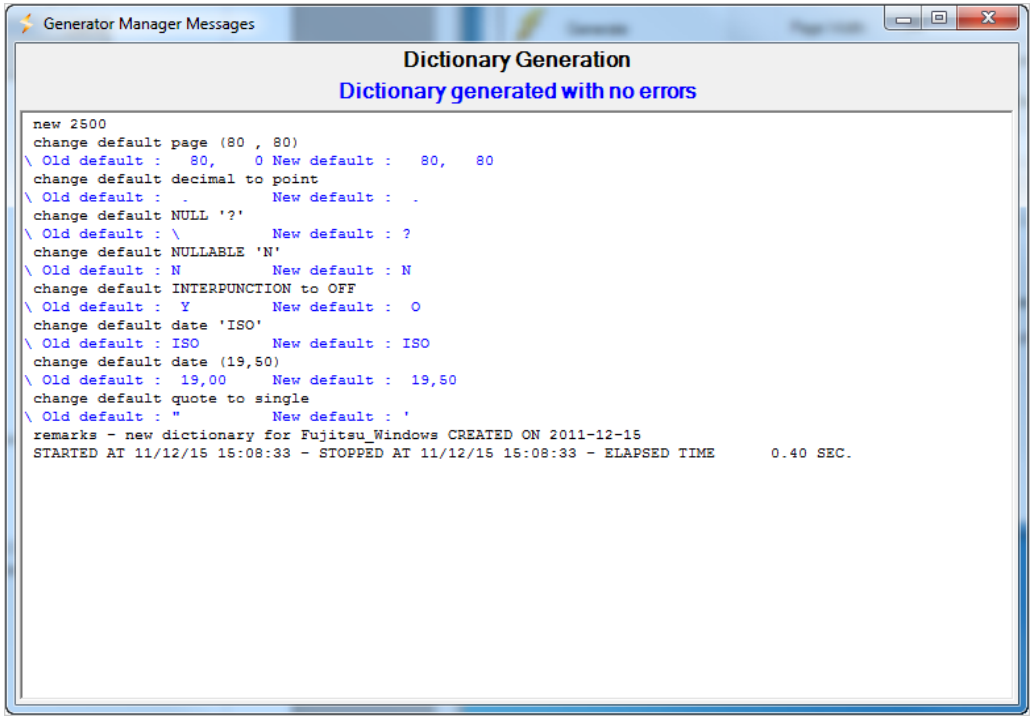
Field	Description
Null character	Enter the single character that will be used to indicate a null value within a field in a sequential file. This *Null Character* will be used for both INBOUND and OUTBOUND NULL storage on a field. You should choose your *Null Character* carefully in case of INBOUND NULL, since the appearance of the *Null Character* on the first position of the field determines that the field has a null value.
Nullable	From the drop-down list, select the default *Nullable* option for fields in a sequential file. There are two options: • Select *N*, if you want all fields with no specific DBNAME setting to be treated as NOT-NULLABLE (i.e. DBNAME 'NOTNULL'). The *Null Character* will not be interpreted for those fields. • Select *I*, if you want all fields with no specific DBNAME setting to be treated as INBOUND-NULLABLE (i.e. DBNAME 'INNULL').
Interpunction	The interpunction option determines the default edit mask for numeric fields. The user can still override this default mask by selecting a specific edit mask for a field in the MetaStore Manager. From the drop-down list, select the required interpunction option. There are three possibilities: • Select *OFF*, if you do not require zero suppressing, nor interpunction. • Select *NO*, if you require zero suppressing, but no interpunction. • Select *YES*, if you require both zero suppressing and interpunction.
Date format	From the drop-down list, select the date format that is to be used when dates are manipulated in an RDBMS environment. The following options are available: • Select *ISO* or *JIS*, if the date components must be separated by a dash (-). • Select *EUR*, if the date components must be separated by a dot (.). • Select *USA*, if the date components must be separated by a slash (/).
SQL Dialect	From the drop-down list, select the SQL Dialect to be used when embedded SQL is generated and no specific SQL DIALECT was defined in the MetaMap Model.
Execution Mode	From the drop-down list, select the required execution mode. The following options are available: • *IMS*: select this option to allow restartability under IMS/DC. • *RESTARTABLE*: select this option to allow restartability for other environments. • Select *(none)* in other situations.
Two-digit Year is Between	This field allows defining the break year, when the data format has no century included like the data format *YYMMDD*.
Language	Reserved for future use.
Import Customized Tables	This option allows to use customized generator tables stored in the DCT folder. Initially, this flag takes the same value as the flag "Use Customized Tables" in the INI Manager, but the User can modify this flag. When activating this option, the User's customizations will automatically be included when generating the Dictionary.

3. Create the Dictionary file.

After having filled out the required fields, click the *Create* button.

The new Dictionary File is generated and overwrites the existing one, if available.

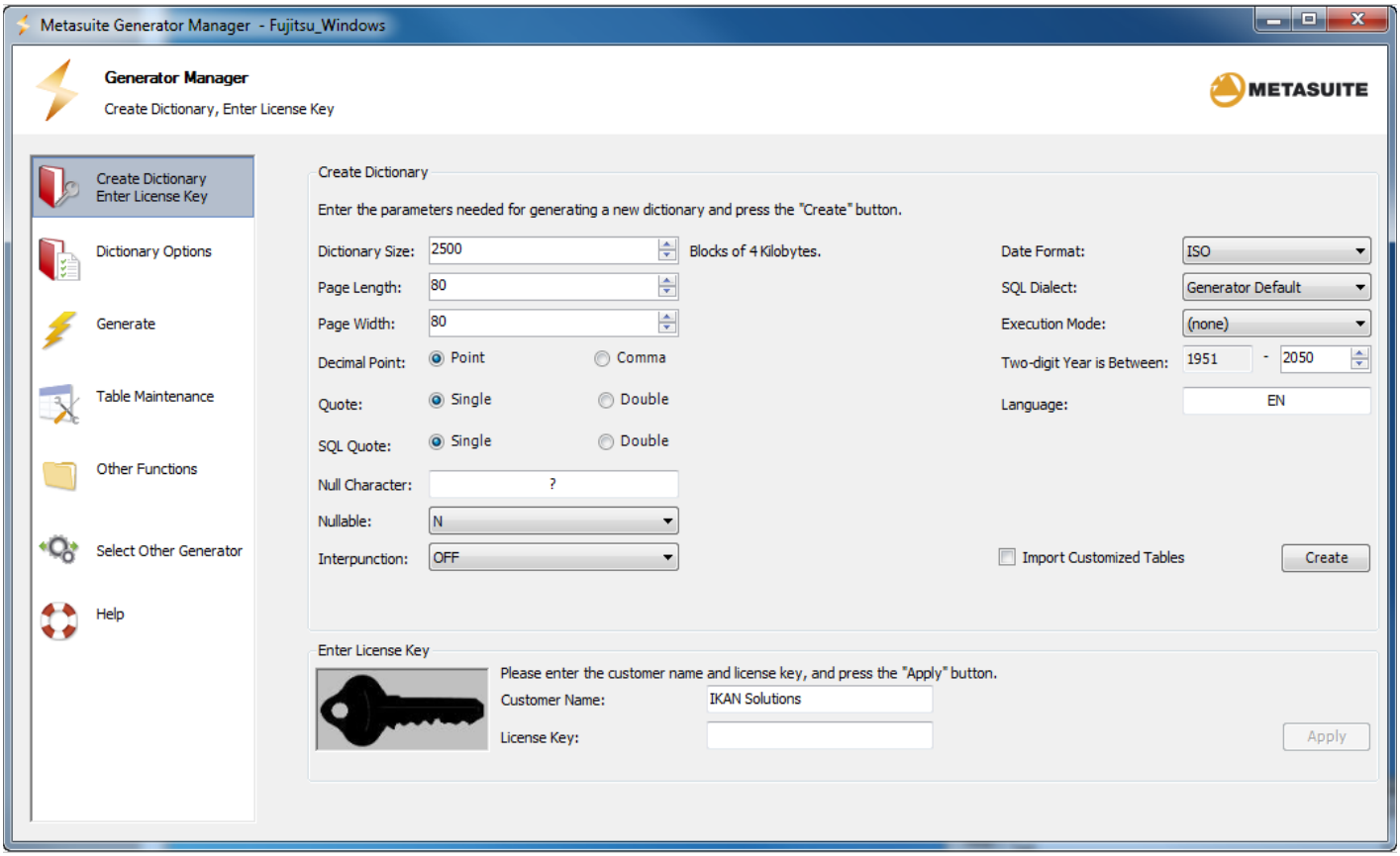
A dialog box listing the old and new settings, as well as any error messages, is displayed.


Generator Manager Messages
Dictionary Generation
Dictionary generated with no errors
new 2500
change default page (80 , 80)
\ Old default : 80, 0 New default : 80, 80
change default decimal to point
\ Old default : . New default : .
change default NULL '?'
\ Old default : \ New default : ?
change default NULLABLE 'N'
\ Old default : N New default : N
change default INTERPUNCTION to OFF
\ Old default : Y New default : O
change default date 'ISO'
\ Old default : ISO New default : ISO
change default date (19,50)
\ Old default : 19,00 New default : 19,50
change default quote to single
\ Old default : " New default : '
remarks - new dictionary for Fujitsu_Windows CREATED ON 2011-12-15
STARTED AT 11/12/15 15:08:33 - STOPPED AT 11/12/15 15:08:33 - ELAPSED TIME 0.40 SEC.

3.2. Entering a License Key

This function allows entering a new License Key. This is required when the old License Key expires. Depending on your contract, you can order a new License Key from IKAN Solutions N.V. or a new License Key will be automatically sent to you by IKAN Solutions N.V.

1. Switch to the *Dictionary/License Key* screen by clicking the appropriate icon in the left column.
   The following screen is displayed:



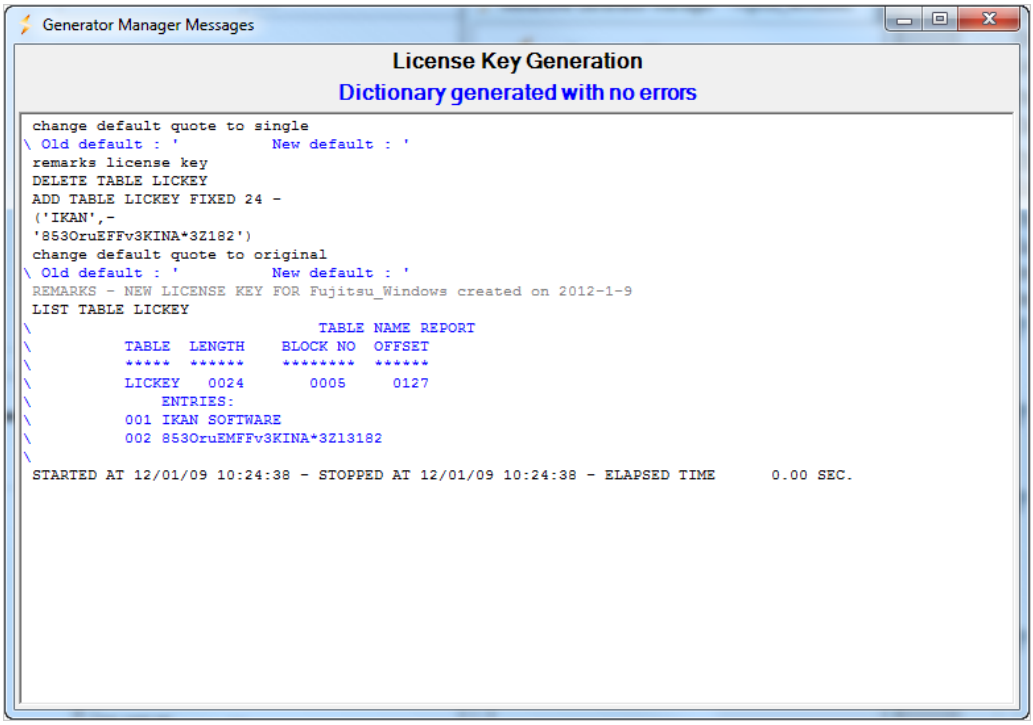
2. Fill out the *Enter License Key* section.
   It contains the following fields:

Field	Description
Customer name	Enter the name of your Organization. Use the exact spelling as provided by IKAN Solutions N.V.
License key	Enter (or paste) the License Key in this field, as provided by IKAN Solutions N.V.

Note: If you use more than one Generator, a license key needs to be generated for each Generator.

3. Generate the License Key.
   After having filled out the required fields, click the *Apply* button.

A dialog box concerning the generation of the License Key is displayed.

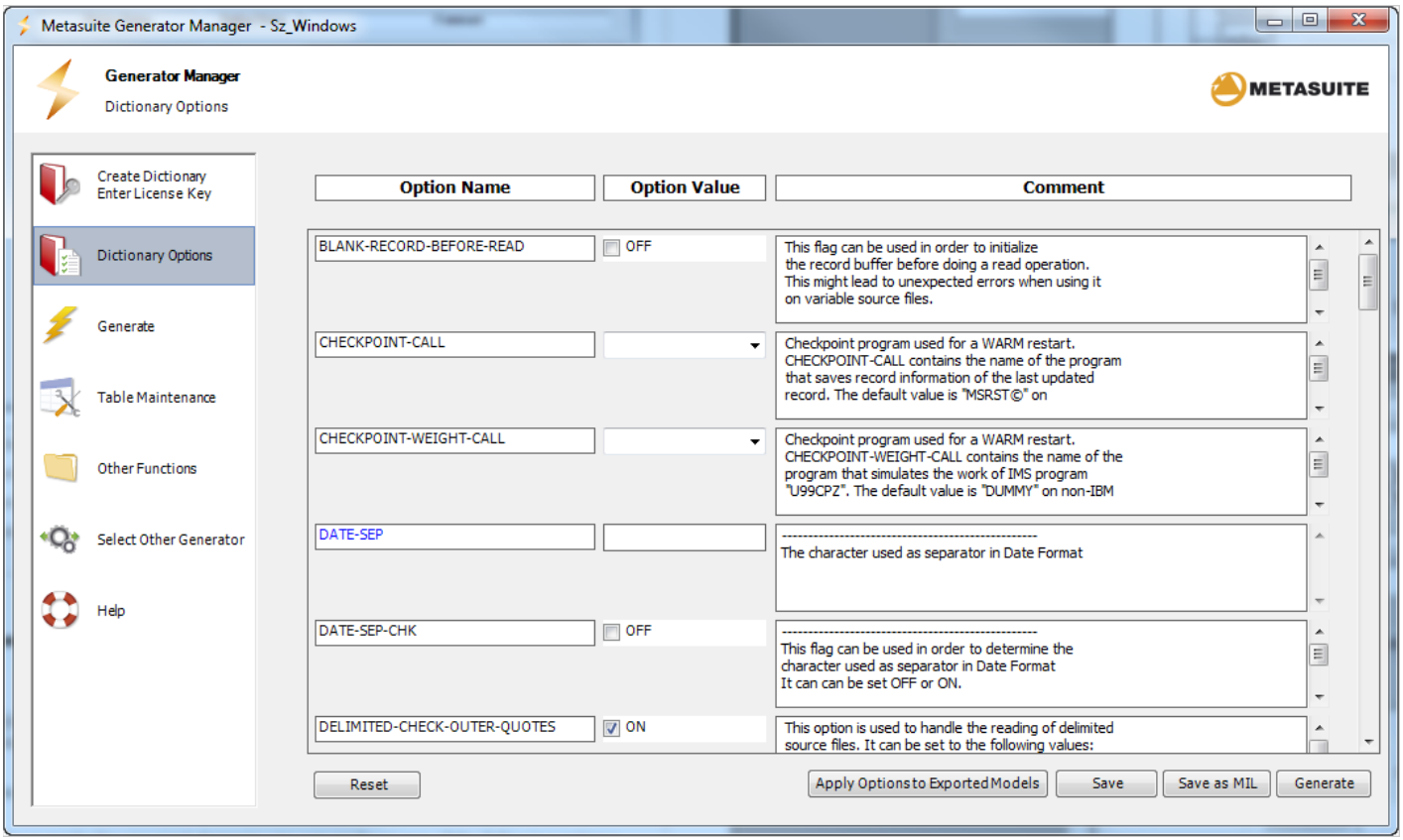

Generator Manager Messages
License Key Generation
Dictionary generated with no errors
change default quote to single
\ Old default : ' New default : '
remarks license key
DELETE TABLE LICKEY
ADD TABLE LICKEY FIXED 24 -
('IKAN',-
'853OruEFFv3KINA*3Z182')
change default quote to original
\ Old default : ' New default : '
REMARKS - NEW LICENSE KEY FOR Fujitsu_Windows created on 2012-1-9
LIST TABLE LICKEY
\ TABLE NAME REPORT
\ TABLE LENGTH BLOCK NO OFFSET
\ ***** ****** ******** ******
\ LICKEY 0024 0005 0127
\ ENTRIES:
\ 001 IKAN SOFTWARE
\ 002 853OruEMFFv3KINA*3Z13182
\
STARTED AT 12/01/09 10:24:38 - STOPPED AT 12/01/09 10:24:38 - ELAPSED TIME 0.00 SEC.

CHAPTER 4
The Dictionary Options Screen

Dictionary Options consist in MTL variables defined in the MTLOPT table. Those options are used in the technical Dictionary tables allowing the generation of the COBOL code (MGL) and the command file (MRL) via the use of the MetaSuite Template Language (page 96).

Users can define their own MTL options by editing the MTLOPT table. You can gain access to this table using the option The Table Maintenance Screen - Managing MGL Tables (page 57). The MTL options must be preceded by comment lines starting with ">* " and the "SET" command must begin immediately after the ">" character on column two. Alphanumeric variables are enclosed by quotes (single or double). Boolean variables can be "Enabled" or "Disabled".

1. Switch to the *Dictionary Options* screen by clicking the appropriate icon in the left column.
   The following screen is displayed:


Metasuite Generator Manager - Sz_Windows
Generator Manager
Dictionary Options
METASUITE
Create Dictionary
Enter License Key
Dictionary Options
Generate
Table Maintenance
Other Functions
Select Other Generator
Help
Option Name
Option Value
Comment
BLANK-RECORD-BEFORE-READ
OFF
This flag can be used in order to initialize the record buffer before doing a read operation. This might lead to unexpected errors when using it on variable source files.
CHECKPOINT-CALL
Checkpoint program used for a WARM restart. CHECKPOINT-CALL contains the name of the program that saves record information of the last updated record. The default value is "MSRST©" on
CHECKPOINT-WEIGHT-CALL
Checkpoint program used for a WARM restart. CHECKPOINT-WEIGHT-CALL contains the name of the program that simulates the work of IMS program "U99CPZ". The default value is "DUMMY" on non-IBM
DATE-SEP
The character used as separator in Date Format
DATE-SEP-CHK
OFF
This flag can be used in order to determine the character used as separator in Date Format It can can be set OFF or ON.
DELIMITED-CHECK-OUTER-QUOTES
ON
This option is used to handle the reading of delimited source files. It can be set to the following values:
Reset
Apply Options to Exported Models
Save
Save as MIL
Generate

For each available option, the following is displayed:

- the *Option Name*
- the *Option Value*
- *Comment* concerning the usage of the option

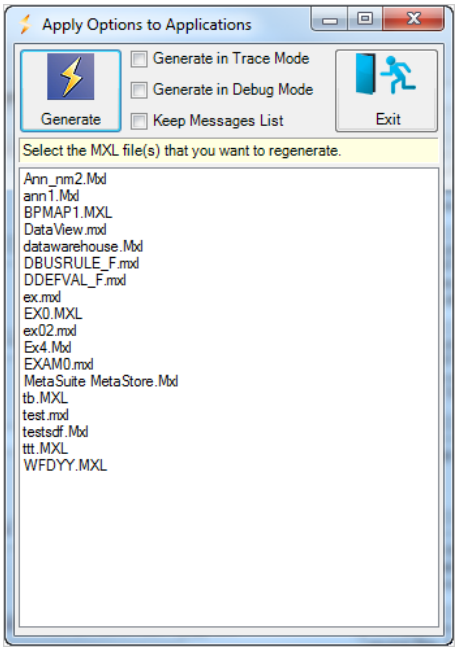
Only the *Option Value* field can be altered. Depending on the option, you may enter a numeric value, an alphanumeric value or switch the activation status.

2. Change the required MTL option settings.

3. Once you have made the required changes, you can perform one of the following actions:

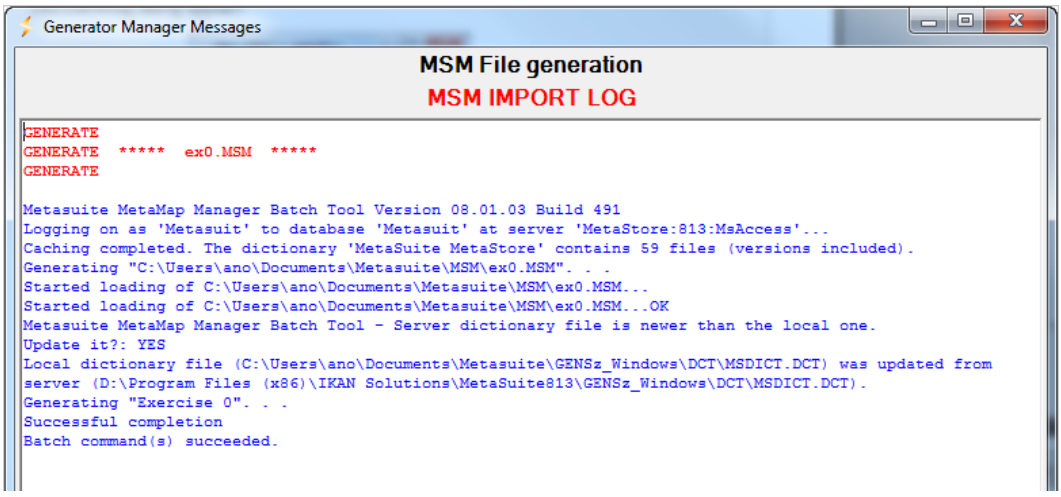
- *Apply Options to Exported Models*

  The following window is displayed:



  Select one or more applications and click the *Generate* button.

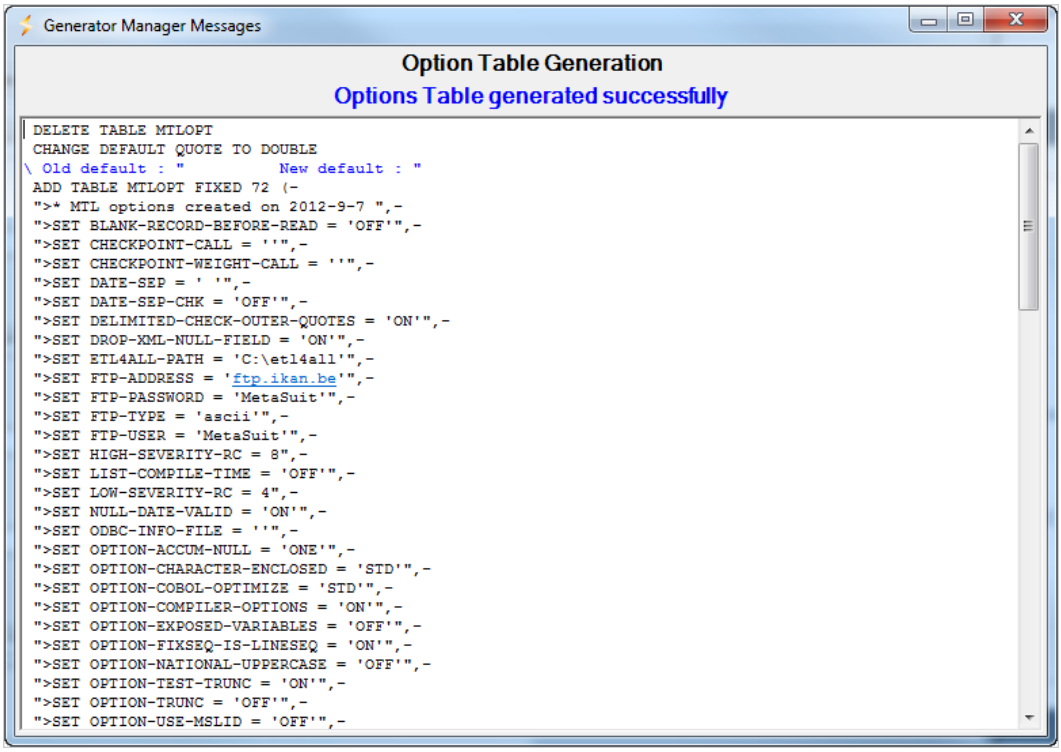
  The *MIL File Generation* window is displayed (if you selected the option *Keep Messages List*):


Generator Manager Messages
MSM File generation
MSM IMPORT LOG
GENERATE
GENERATE ***** ex0.MSM *****
GENERATE
Metasuite MetaMap Manager Batch Tool Version 08.01.03 Build 491
Logging on as 'Metasuit' to database 'Metasuit' at server 'MetaStore:813:MsAccess'...
Caching completed. The dictionary 'MetaSuite MetaStore' contains 59 files (versions included).
Generating "C:\Users\ano\Documents\Metasuite\MSM\ex0.MSM". . .
Started loading of C:\Users\ano\Documents\Metasuite\MSM\ex0.MSM...
Started loading of C:\Users\ano\Documents\Metasuite\MSM\ex0.MSM...OK
Metasuite MetaMap Manager Batch Tool - Server dictionary file is newer than the local one.
Update it?: YES
Local dictionary file (C:\Users\ano\Documents\Metasuite\GENSz_Windows\DCT\MSDICT.DCT) was updated from
server (D:\Program Files (x86)\IKAN Solutions\MetaSuite813\GENSz_Windows\DCT\MSDICT.DCT).
Generating "Exercise 0". . .
Successful completion
Batch command(s) succeeded.

- *Generate*

  Generating the MTL option settings will implement them immediately in the Generator Dictionary.

The following window is displayed:


Generator Manager Messages
Option Table Generation
Options Table generated successfully
DELETE TABLE MTLOPT
CHANGE DEFAULT QUOTE TO DOUBLE
\ Old default : " New default : "
ADD TABLE MTLOPT FIXED 72 (-
">* MTL options created on 2012-9-7 ",-
">SET BLANK-RECORD-BEFORE-READ = 'OFF'",-
">SET CHECKPOINT-CALL = ''",-
">SET CHECKPOINT-WEIGHT-CALL = ''",-
">SET DATE-SEP = ' '",-
">SET DATE-SEP-CHK = 'OFF'",-
">SET DELIMITED-CHECK-OUTER-QUOTES = 'ON'",-
">SET DROP-XML-NULL-FIELD = 'ON'",-
">SET ETL4ALL-PATH = 'C:\etl4all'",-
">SET FTP-ADDRESS = 'ftp.ikan.be'",-
">SET FTP-PASSWORD = 'MetaSuit'",-
">SET FTP-TYPE = 'ascii'",-
">SET FTP-USER = 'MetaSuit'",-
">SET HIGH-SEVERITY-RC = 8",-
">SET LIST-COMPILE-TIME = 'OFF'",-
">SET LOW-SEVERITY-RC = 4",-
">SET NULL-DATE-VALID = 'ON'",-
">SET ODBC-INFO-FILE = ''",-
">SET OPTION-ACCUM-NULL = 'ONE'",-
">SET OPTION-CHARACTER-ENCLOSED = 'STD'",-
">SET OPTION-COBOL-OPTIMIZE = 'STD'",-
">SET OPTION-COMPILER-OPTIONS = 'ON'",-
">SET OPTION-EXPOSED-VARIABLES = 'OFF'",-
">SET OPTION-FIXSEQ-IS-LINESEQ = 'ON'",-
">SET OPTION-NATIONAL-UPPERCASE = 'OFF'",-
">SET OPTION-TEST-TRUNC = 'ON'",-
">SET OPTION-TRUNC = 'OFF'",-
">SET OPTION-USE-MSLID = 'OFF'",-

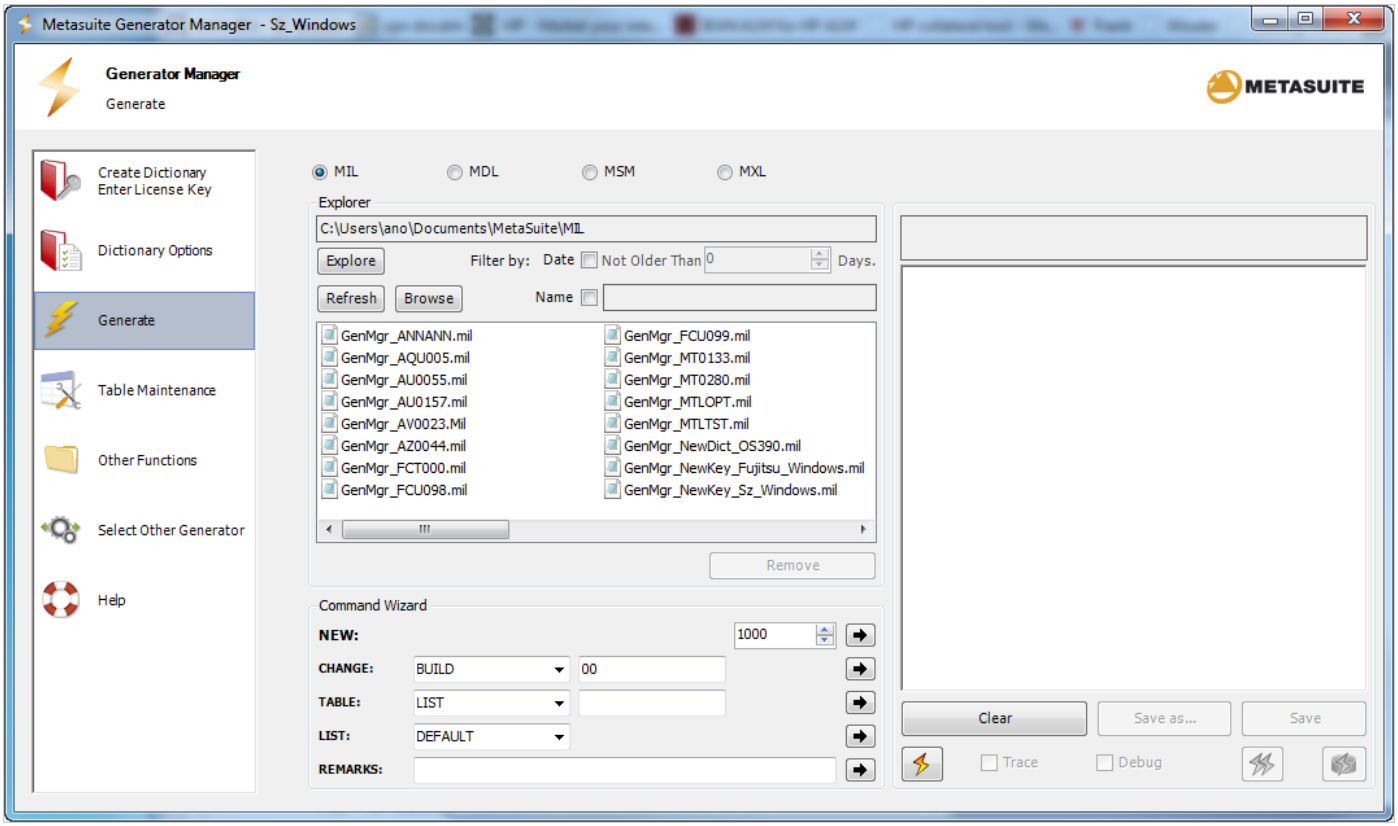
CHAPTER 5

The Generate Screen - Overview

The *Generate* screen allows you to generate the different file types: MIL, MDL, MSM and MXL. Refer to the dedicated chapters for more specific information:

- The Generate Screen - Implementing MIL Instructions (page 18)
- The Generate Screen - Working With MDL Files (page 36)
- The Generate Screen - Working With MSM Files (page 46)
- The Generate Screen - Working With MXL Files (page 48)

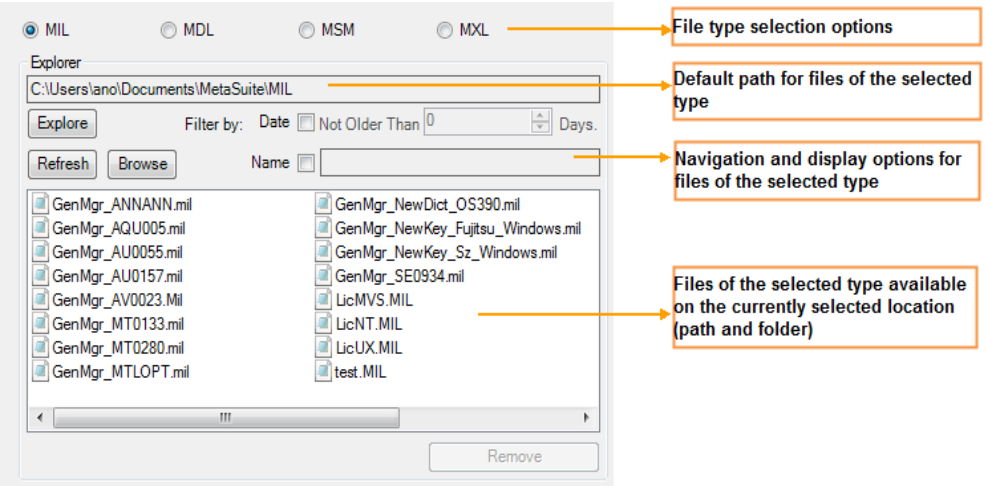
1. Switch to the *Generate* screen by clicking the appropriate icon in the left column.
   The following screen is displayed:



The common elements available are listed in the upper-left corner:

- File types
- Explorer options
- Command Wizard
- Code Viewer

- Possible actions



2. Select the required file type.

   The following types are available:

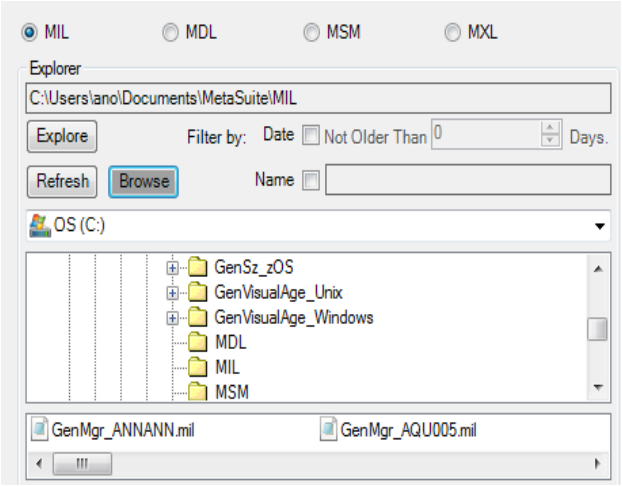
   - MIL (page 18)
   - MDL (page 36)
   - MSM (page 46)
   - MXL (page 48)

3. Verify the path.

   The indicated path is the default path for the selected file type.

4. If you do not immediately find the file you require in the list of available files, you can use the *Explorer* options:

Option	Description
Explore	Select this option to open Windows Explorer. Browse to another folder if required. You can also copy, edit or delete available files.
Refresh	This option allows refreshing the list of available files in the selected folder. It may be useful to select this option, if the file you expect is not displayed.

Option	Description
Browse	This option allows selecting another folder containing files of the selected type: • Click the *Browse* button. The hierarchical structure of the folder is displayed:  • Browse to the folder containing the required files. • Click the *Browse* button again to hide the hierarchy.
Filter by Date	Select this check box, if you want to define a filter based on a date. Enter the number of days in the *Not older than* field and hit return. For example: if you enter *30*, the system will search for all available files not older than 30 days.
Filter by Name	Select this check box, if you want to define a filter based on the file name. Enter a character string. The list of available files will be limited to files whose name contains the defined string. Clear the check box to redisplay all available files.

CHAPTER 6

The Generate Screen - Implementing MIL Instructions

MIL stands for MetaSuite Installation Language. This option allows you to execute a instructions. You can set up this list in three ways:

1. Copying Existing MIL Files (page 20)
2. Adding MIL Instructions Using the Command Wizard (page 21)
3. Entering MIL instructions manually

When executing the MIL instructions, the referenced parameters in the Dictionary File are updated. You do not have to create the complete dictionary anew. The settings that were not touched by the executed MIL instructions remain identical.

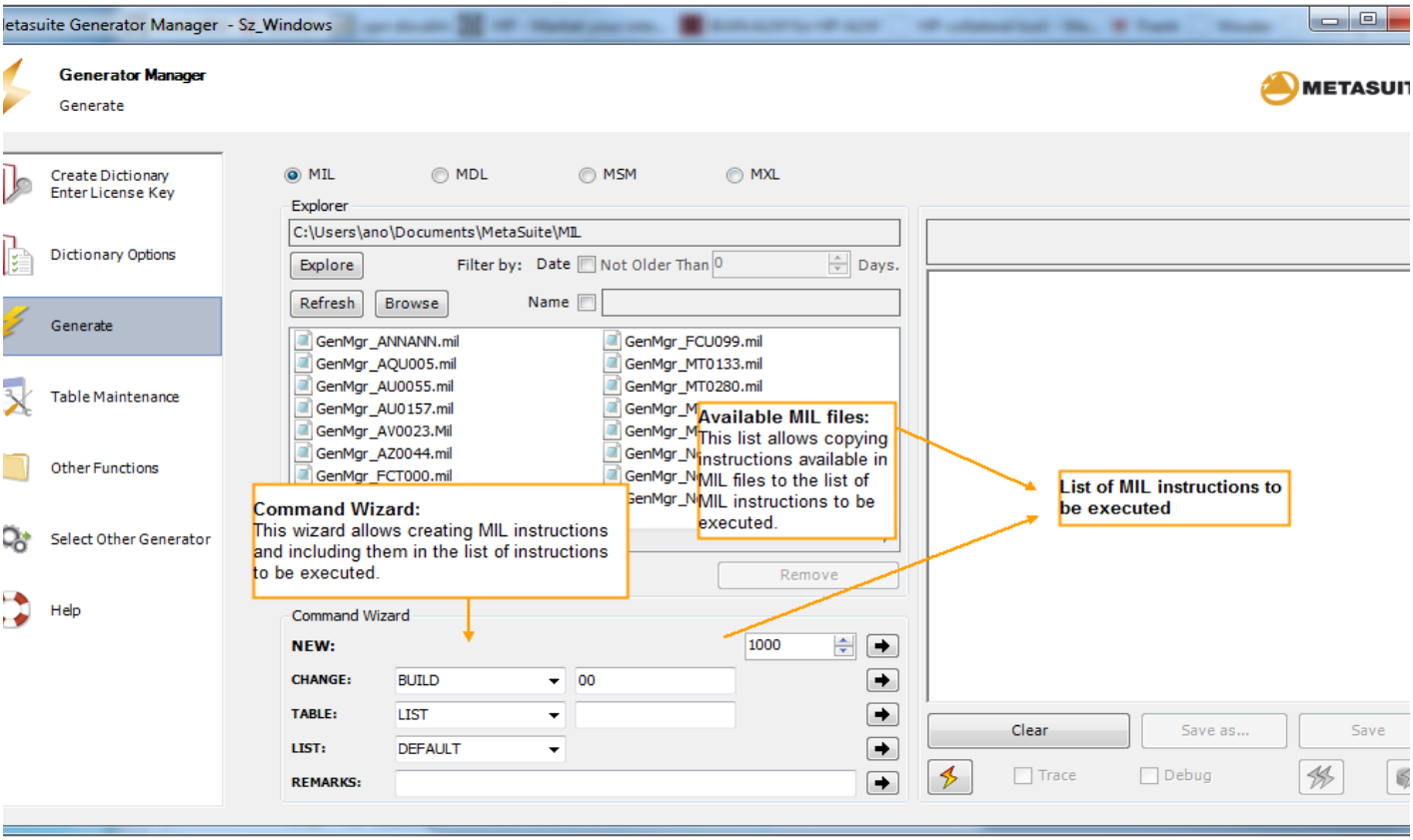
This chapter also contains the following reference sections:

- CHANGE Options Reference (page 24)
- TABLE Options Reference (page 32)

6.1. The Generate MIL Screen

1. Click the *Generate* icon in the left column.

2. Select the *MIL* format option.
   The following screen is displayed:



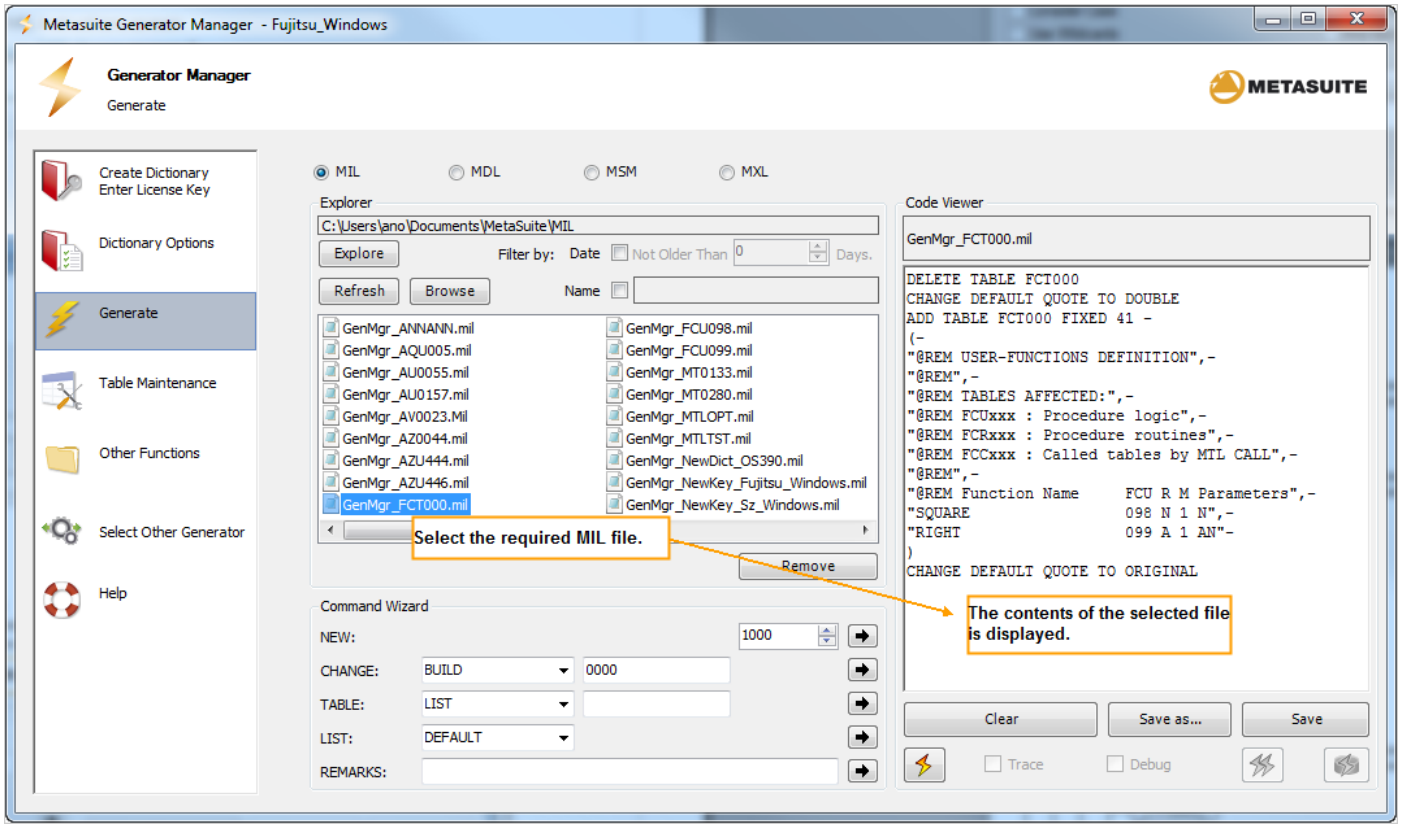
The MIL instructions dialog consists of three sections:

Section	Description
List of available MIL files	This list is displayed in the upper-left corner. Select a file to add its contents to the list of MIL instructions to be executed on the right. See Copying Existing MIL Files on page 20.
Command Wizard	Use the Command Wizard to manually create MIL instructions in order to include them in the list of MIL instructions to be executed. See Adding MIL Instructions Using the Command Wizard on page 21.
List of MIL instructions to be executed	This list contains MIL instructions that were added from a MIL file and/or from the Command Wizard. You can manually edit the listed instructions or enter additional instructions to the list.

6.2. Copying Existing MIL Files

1. Switch to the MIL instructions screen.
   See The Generate MIL Screen on page 19.

2. Select the required MIL file in the list of available MIL Files.
   Its content is immediately displayed in the list of MIL instructions to be implemented:

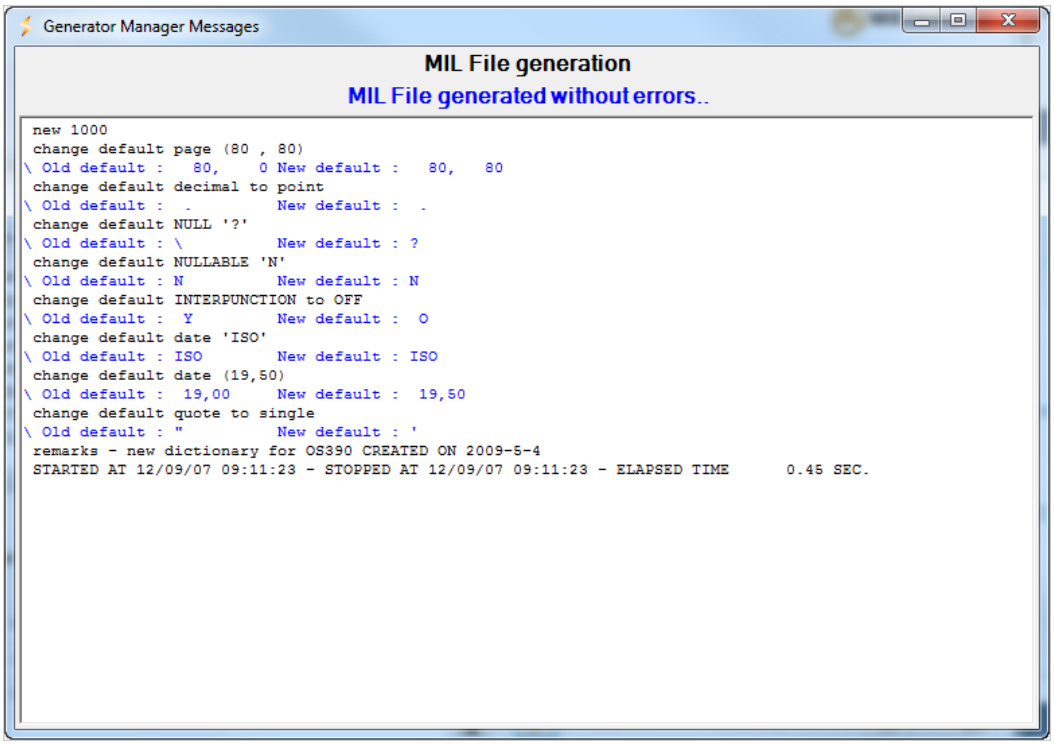


Note: If you select another MIL file now, the content of this file replaces the instructions already displayed. You can delete a MIL file, by selecting it in the list of available MIL files and then clicking *Remove* at the screen bottom.

3. Edit the displayed MIL commands if required.
   You can also use the Command Wizard to insert commands. See Adding MIL Instructions Using the Command Wizard on page 21.

4. Once you have made the required changes, you can perform one of the following actions:

Action	Description
Clear	Clears the list content so that you can start over again.
Save as	Saves the displayed MIL commands in a new MIL file. Enter the new Path and File name and click *OK*.

Action	Description
Save	Saves the displayed (edited) MIL commands in the existing MIL File.
	Click the *Generate* button if you want to implement the displayed MIL commands. The MIL commands will be implemented, so that the new settings become valid in the Dictionary File. The following dialog confirms this action: 

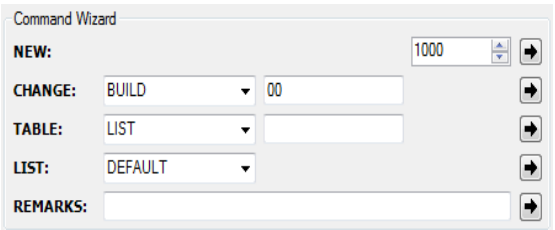
Generator Manager Messages
MIL File generation
MIL File generated without errors..
new 1000
change default page (80 , 80)
\ Old default : 80, 0 New default : 80, 80
change default decimal to point
\ Old default : . New default : .
change default NULL '?'
\ Old default : \ New default : ?
change default NULLABLE 'N'
\ Old default : N New default : N
change default INTERPUNCTION to OFF
\ Old default : Y New default : O
change default date 'ISO'
\ Old default : ISO New default : ISO
change default date (19,50)
\ Old default : 19,00 New default : 19,50
change default quote to single
\ Old default : " New default : '
remarks - new dictionary for OS390 CREATED ON 2009-5-4
STARTED AT 12/09/07 09:11:23 - STOPPED AT 12/09/07 09:11:23 - ELAPSED TIME 0.45 SEC.

6.3. Adding MIL Instructions Using the Command Wizard

1. Switch to the MIL instructions screen.
   See The Generate MIL Screen on page 19.

Note: If required, select a MIL file in the list of available MIL Files. See Copying Existing MIL Files on page 20.

The Command Wizard is located in the lower left corner:

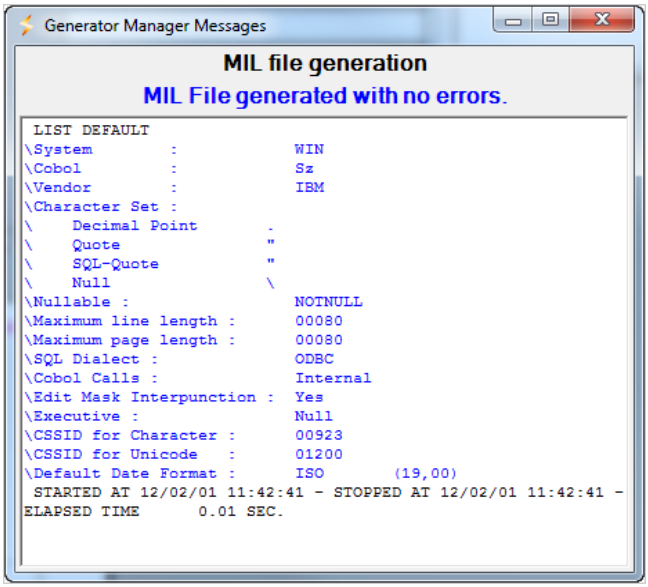
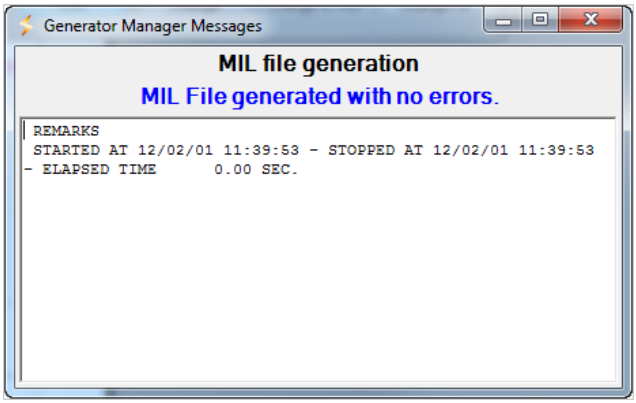


Each line represents a category of MIL instructions.

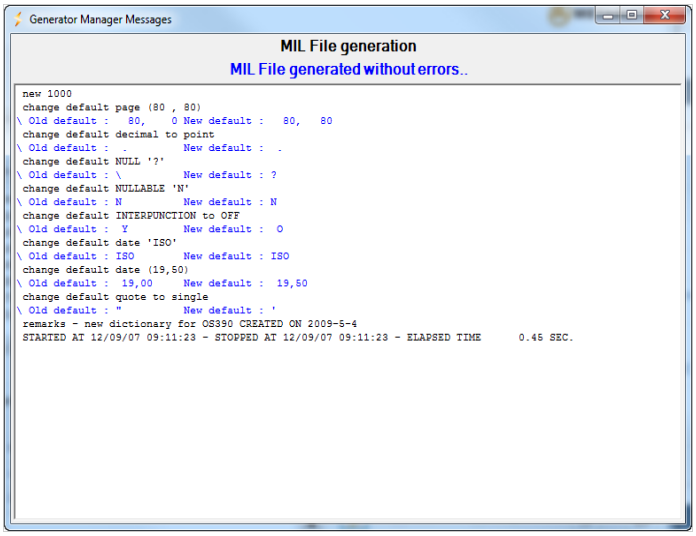
2. Select the required instructions and click the ➔ button to add them to the MIL Instructions list to the right.

The following categories are available:

Category	Description
NEW	This Category allows the insertion of a *NEW* command in the list of MIL commands to be implemented. The *NEW* command creates a new Generator library. It must be the first command, when you perform the initial load of a new Generator library. In the numeric field to the right: 1. Enter the four-digit number representing the number of blocks to be initialized for the new Generator library 2. Click the arrow button
CHANGE	To add a CHANGE instruction: 1. Select the element to be changed from the drop-down list 2. Select the element value 3. Click the arrow button The following options are available: • BUILD (page 25) • DATE FORMAT (page 25) • DATE BREAK YEAR (page 26) • DECIMAL (page 26) • EXEC (page 27) • INTERPUNCTION (page 27) • NULL (page 28) • NULLABLE (page 28) • PAGE (page 29) • QUOTE (page 29) • SQL (page 30) • SQL-QUOTE (page 30) • DYNAMIC (page 31) • LANGUAGE (page 31) • UNICODE-CCSID (page 31) • CHARACTER-CCSID (page 32) Refer to the indicated sections for detailed information.
TABLE	To add a TABLE instruction: 1. Select the table action type from the drop-down list 2. Enter the name of the table to be listed 3. Click the arrow button The following actions are available: • LIST (page 33) • ADD (page 33) • COPY (page 34) • DELETE (page 34) • IMPORT (page 35) • EXPORT (page 35) Refer to the indicated sections for detailed information.

Category	Description
LIST	The following options are available: • DEFAULT Select DEFAULT to display a list of all the defaults that apply to the Generator Library. The following type of information will be displayed:  • VERSION Select VERSION to list the version and build number of the current generator and generator library. The output of this command contains also the build number requirements of the generator and the Generator library: a certain build of a generator corresponds with a certain build number of the dictionary, and vice versa. The following type of information will be displayed: 
REMARKS	To add a REMARK to the list of MIL instructions: 1. In the text field, enter the remarks you want to add 2. Click the arrow button

3. Once you have added the required MIL instructions, you can perform one of the following actions:

Action	Description
Clear	Clears the list content so that you can start over again.
Save as	Saves the displayed MIL commands in a new MIL file. Enter the new Path and File name and click *OK*.
Save	Saves the displayed (edited) MIL commands in the existing MIL File.
	Click the *Generate* button if you want to implement the displayed MIL commands. The MIL commands will be implemented, so that the new settings become valid in the Dictionary File. The following dialog confirms this action: 

Generator Manager Messages
MIL File generation
MIL File generated without errors..
new 1000
change default page (80 , 80)
\ Old default : 80, 0 New default : 80, 80
change default decimal to point
\ Old default : . New default : .
change default NULL '?'
\ Old default : \ New default : ?
change default NULLABLE 'N'
\ Old default : N New default : N
change default INTERPUNCTION to OFF
\ Old default : Y New default : O
change default date 'ISO'
\ Old default : ISO New default : ISO
change default date (19,50)
\ Old default : 19,00 New default : 19,50
change default quote to single
\ Old default : " New default : '
remarks - new dictionary for OS390 CREATED ON 2009-5-4
STARTED AT 12/09/07 09:11:23 - STOPPED AT 12/09/07 09:11:23 - ELAPSED TIME 0.45 SEC.

6.4. CHANGE Options Reference

This section contains reference information about the following CHANGE options:

- BUILD (page 25)
- DATE FORMAT (page 25)
- DATE BREAK YEAR (page 26)
- DECIMAL (page 26)
- EXEC (page 27)
- INTERPUNCTION (page 27)
- NULL (page 28)
- NULLABLE (page 28)
- PAGE (page 29)
- QUOTE (page 29)
- SQL (page 30)
- SQL-QUOTE (page 30)
- DYNAMIC (page 31)

- LANGUAGE (page 31) (reserved for future use)
- UNICODE-CCSID (page 31)
- CHARACTER-CCSID (page 32)

For a detailed procedure description, refer to the following sections:

- Copying Existing MIL Files (page 20)
- Adding MIL Instructions Using the Command Wizard (page 21)

BUILD

The *CHANGE DEFAULT BUILD* command allows changing the Dictionary Build Number after applying a patch. Thanks to this command, the generation of a new dictionary can be avoided.

1. On the *CHANGE* line in the Command Wizard, select *BUILD* from the drop-down.

2. Enter the Dictionary Build Number in the numeric field to the right.
   This build number is mandatory and must be between 00 and 99.

3. Click the arrow button to the right to add the *CHANGE DEFAULT BUILD* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

DATE FORMAT

The *CHANGE DEFAULT DATE* command allows selecting the date format that is to be used when dates are manipulated in an RDBMS environment.

1. On the *CHANGE* line in the Command Wizard, select *DATE FORMAT* from the drop-down list.
   A new drop-down list containing the available date format options appears on the right.

2. Select the required date format option.

Option	Format
ISO	YYYY-MM-DD
EUR	DD.MM.YYYY
JIS	YYYY-MM-DD
USA	MM/DD/YYYY

Note: The date format of source data in an RDBMS environment must be set to ISO. This can be done via a precompiler or a compiler option. The date format of source data from a flat file is defined in the MetaStore Dictionary. The delimiter between character dates is unimportant.

3. Click the arrow button to the right to add the *CHANGE DEFAULT DATE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

DATE BREAK YEAR

The *CHANGE DATE BREAK YEAR* command allows determining whether years are representing the 20th or the 21st century when the date format has no century included such as the date format *YYMMDD*.

1. On the *CHANGE* line in the Command Wizard, select *DATE BREAK YEAR* from the drop-down list.

2. Specify the century and break year for date indications where years are specified as a 2-digit number.

   The combination of Break Year and Break Century is used for converting 2-digit years into 4-digit years. The combination of the two parameters form the oldest (lowest) year a 2-date year can be converted to.

   - In the first field, enter the lowest century.
   - In the second field, enter the year break point.

   For example:

   If you specify (19 , 60) as break century and break year, 1960 will be the oldest possible year.

   - The year indications from *00* to *59* are interpreted as *2000* to *2059*.
   - The year indications from *60* to *99* are interpreted as *1960* to *1999*.

3. Click the arrow button to the right to add the *CHANGE DATE BREAK YEAR* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

DECIMAL

The *CHANGE DEFAULT DECIMAL* command allows selecting the required Decimal Point character. You can choose between *Comma* and *Point*.

1. On the *CHANGE* line in the Command Wizard, select *DECIMAL* from the drop-down list.

2. Select *Point* or *Comma*.

3. Click the arrow button to the right to add the *CHANGE DEFAULT DECIMAL* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

EXEC

The *CHANGE DEFAULT EXEC* command allows selecting the required Execution Mode.

1. On the *CHANGE* line in the Command Wizard, select *EXEC* from the drop-down list.
   A second drop-down menu is displayed to the right.

2. Select the required execution mode.

Execution Mode	Explanation
(none)	Select *(none)* in all other situations.
IMS	This option allows restartability under IMS/DC.
RESTARTABLE	This option allows restartability for other environments.

3. Click the arrow button to the right to add the *CHANGE DEFAULT EXEC* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

INTERPUNCTION

The *CHANGE DEFAULT INTERPUNCTION* command is used to specify the default edit mask for numeric values.

1. On the *CHANGE* line in the Command Wizard, select *INTERPUNCTION* from the drop-down list.
   A second drop-down menu is displayed to the right.

2. Select the required interpunction option.

Interpunction	Explanation
OFF	Every numeric field must be treated as a code, i.e. no zero compression and no interpunction.
NO	The default edit mask for numeric fields (non-code) must have zero compression but no interpunction. Note that code numerics will have zero suppression nor interpunction.
YES	The default edit mask for numeric fields (non-code) must have both zero suppression and interpunction. Note that code numerics will have zero suppression nor interpunction.

3. Click the arrow button to the right to add the *CHANGE DEFAULT INTERPUNCTION* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

NULL

The *CHANGE DEFAULT NULL* command allows defining the single character that will be used to indicate a null value within a field in a sequential file.

This null character will be used for both INBOUND and OUTBOUND NULL storage on a field.

You should choose your null character carefully in case of INBOUND NULL, since the appearance of the null character on the first position of the field determines that the field has a null value.

1. On the *CHANGE* line in the Command Wizard, select *NULL* from the drop-down list.
   A text field containing the *?* character, appears on the right.

2. Replace this character by the required null value character.

3. Click the arrow button to the right to add the *CHANGE DEFAULT NULL* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

NULLABLE

The *CHANGE DEFAULT NULLABLE* command allows defining the default NULLABLE option for fields.

1. On the *CHANGE* line in the Command Wizard, select *NULLABLE* from the drop-down list.
   A new drop-down list appears on the right.

2. Select the required option.

Option	Explanation
N	All fields without specific DBNAME setting will be treated as NOT-NULLABLE (i.e. DBNAME 'NOTNULL'). The Null Character will not be interpreted for those fields.
I	All fields with no specific DBNAME setting will be treated as INBOUND-NULLABLE (i.e. DBNAME 'INNULL').

3. Click the arrow button to the right to add the *CHANGE DEFAULT NULLABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

PAGE

The *CHANGE DEFAULT PAGE* command allows defining the default page length (in number of lines) and page width (in number of characters) that is used for target reports when no page settings are specified by the MetaMap model.

1. On the *CHANGE* line in the Command Wizard, select *PAGE* from the drop-down list.
   Two text fields appear on the right.

2. Enter a new default value for the Page Length in the first text box.

3. Enter a new default value for the Page Width in the second text box.

4. Click the arrow button to the right to add the *CHANGE DEFAULT PAGE* command in the command window.

5. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

QUOTE

The *CHANGE DEFAULT QUOTE* command allows defining the required quote character. You can choose between *Single* and *Double*.

This quote character is used for surrounding table sentences in the MIL files and for surrounding COBOL literals in the MGL files.

The main use of the *CHANGE DEFAULT QUOTE* command is to temporarily change the default quote character to a double quote, prior to adding or changing one or more table definitions. Changing the default quote character before entering the *ADD TABLE* command allows you to embed single quote characters in the text of any table entry being added or replaced. After all of the table commands have been entered, you should then change the default back to a single quote by coding a second *CHANGE DEFAULT QUOTE* command.

1. On the *CHANGE* line in the Command Wizard, select *QUOTE* from the drop-down list.
   You can choose between Single and Double quotes. Two option buttons appear on the right.

2. Select *Single* or *Double*.

3. Click the arrow button to the right to add the *CHANGE DEFAULT QUOTE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

SQL

The *CHANGE DEFAULT SQL DIALECT* command allows defining the SQL Dialect to be used when embedded SQL is generated and no specific SQL Dialect was defined in the MetaMap Model.

1. On the *CHANGE* line in the Command Wizard, select *SQL* from the drop-down list.
   A new drop-down list appears on the right, containing the following options:
   - DB2 for OS/400
   - DB2 for z/OS
   - DB2 LUW
   - DB2/2
   - DB2/VSE
   - Generator Default
   - Informix
   - Ingres
   - MySQL
   - ODBC
   - Oracle
   - Oracle/RDB
   - SESAM
   - SQL Server
   - Sybase
   - Teradata

2. Select the required SQL dialect.

3. Click the arrow button to the right to add the *CHANGE DEFAULT SQL DIALECT* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

SQL-QUOTE

The *CHANGE DEFAULT SQL-QUOTE* command modifies the quote setting for the SQL commands in the generated COBOL programs.

1. On the *CHANGE* line in the Command Wizard, select *SQL-QUOTE* from the drop-down list.
   You can choose between Single and Double quotes.

2. Select *Single* or *Double*.

3. Click the arrow button to the right to add the *CHANGE DEFAULT SQL-QUOTE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

DYNAMIC

The *CHANGE DEFAULT DYNAMIC* command describes how the generated MetaSuite program calls the runtime modules.

1. On the *CHANGE* line in the Command Wizard, select *DYNAMIC* from the drop-down list.
   The possible values are:
   - I = internal COBOL calls (the runtime modules are part of the generated COBOL source)
   - Y = dynamic COBOL calls
   - N = static COBOL calls

2. Select the required option.

3. Click the arrow button to the right to add the *CHANGE DEFAULT DYNAMIC* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

LANGUAGE

This name is a two-character abbreviation of the language of the generator messages.
Currently the languages English (EN) and French (FR) are supported.

1. On the *CHANGE* line in the Command Wizard, select *LANGUAGE* from the drop-down list.

2. Enter the required language abbreviation: EN or FR.

3. Click the arrow button to the right to add the *CHANGE DEFAULT LANGUAGE TO* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

UNICODE-CCSID

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. Per language, country or region, a specific set of characters is used to define a regional "Code Page".

Unicode character sets should have escaped from this kind of language specific behavior, but they did not. The first 256 characters of the Unicode character set are CCSID dependent.

The *CHANGE DEFAULT UNICODE-CCSID* command defines the standard CCSID for double-byte character sets.

1. On the *CHANGE* line in the Command Wizard, select *UNICODE-CCSID* from the drop-down list.
2. Enter the required value.
3. Click the arrow button to the right to add the *CHANGE DEFAULT UNICODE-CCSID* command in the command window.
4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

CHARACTER-CCSID

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. Per language, country or region, a specific set of characters is used to define a regional "Code Page".

The *CHANGE DEFAULT CHARACTER-CCSID* command defines the standard CCSID for single-byte character sets.

1. On the *CHANGE* line in the Command Wizard, select *CHARACTER-CCSID* from the drop-down list.
2. Enter the required value.
3. Click the arrow button to the right to add the *CHANGE DEFAULT CHARACTER-CCSID* command in the command window.
4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

6.5. TABLE Options Reference

This section contains reference information about the following TABLE options:

- LIST (page 33)
- ADD (page 33)
- COPY (page 34)
- DELETE (page 34)
- IMPORT (page 35)
- EXPORT (page 35)

LIST

The *LIST TABLE* command allows listing a selected table or all tables available in the Generator library.

This is particularly useful to produce a hard copy listing of each Generator library table before modifying or deleting it, in the event that it becomes necessary to restore the table to its original state.

1. On the *TABLE* line in the Command Wizard, select *LIST* from the drop-down list.

2. Enter the name of the Table to be listed in the text box to the right of the drop-down list.

   If you leave this field blank or enter *ALL* in this field, all Tables will be listed.

3. Click the arrow button to the right to add the *LIST TABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

ADD

The *ADD TABLE* command allows adding a table to the Generator library.

There are four basic types of library tables:

- System Tables

  These tables are used for reserved words, error messages, and tables of system-dependent parameters.

- COBOL code tables

  These tables are the building blocks of the COBOL source.

- Code-control Tables

  These tables are defined in conjunction with the *CODE-CONTROL* option in MetaMap. The user can add self-defined code-control Tables to the Dictionary.

- File Code Tables

  These tables contain a string of 32 characters. Each character represents an option. File Control Tables are becoming obsolete. They are replaced by code-control Tables.

This option is mainly used to define code-control Tables and their associated Prototype Code Tables.

1. On the *TABLE* line in the Command Wizard, select *ADD* from the drop-down list.

2. In the first text box to the right of the drop-down list, enter the name of the table being defined to the library.

   The name must

   - be unique within the library,
   - may be up to 6 characters in length,
   - must begin with an alphabetic character,
   - may contain the characters A-Z, 0-9 and embedded hyphens

3. In the second text box to the right of the drop-down list, enter the maximum length, in number of characters, of any one entry in the table.

   Any entries that are shorter than the defined value will be padded on the right with blanks. The maximum entry size should be no longer than 72 for COBOL code-control Tables (MGL). For other tables, the maximum value is 99.

4. Click the arrow button to the right to add the command in the command window.

5. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

COPY

The *COPY TABLE* command is used to copy the content of the selected Generator library table to the *mscopy.mil* output file, which is saved in the following folder:

<InstallationFolder>\GENXXX\TMP

where *XXX* is the indication of the selected generator.

1. On the *TABLE* line in the Command Wizard, select *COPY* from the drop-down list.

2. In the text box to the right of the drop-down list, enter the name of the table to be copied to the *mscopy.mil* file.

3. Click the arrow button to the right to add the *COPY TABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

DELETE

The *DELETE TABLE* command is used to delete a table from the Generator library.

Be careful not to delete any of the System Tables from the library, as this may render the system unusable. You can ensure your ability to recover from such an error by using the COPY TABLE command (to copy the definition of the table to be deleted to a backup file) prior to executing the DELETE command.

1. On the *TABLE* line in the Command Wizard, select *DELETE* from the drop-down list.

2. In the text box to the right of the drop-down list, enter the name of the table to be deleted.

3. Click the arrow button to the right to add the *DELETE TABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

IMPORT

The *IMPORT TABLE* command is used to copy the content of a text file, named *xxxxxx.tbl* from the DCT folder to the Generator Dictionary. The table name should be 6 characters long. "ALL" is not permitted.

1. On the *TABLE* line in the Command Wizard, select *IMPORT* from the drop-down list.

2. In the text box to the right of the drop-down list, enter the name of the table to be imported.

3. Click the arrow button to the right to add the *IMPORT TABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

EXPORT

The *EXPORT TABLE* command is used to copy the content of the Generator Dictionary to a text file, named *xxxxxx.tbl* in the DCT folder. The table name should be 6 characters long. "ALL" is not permitted.

1. On the *TABLE* line in the Command Wizard, select *EXPORT* from the drop-down list.

2. In the text box to the right of the drop-down list, enter the name of the table whereto you want export the Generator Dictionary.

3. Click the arrow button to the right to add the *EXPORT TABLE* command in the command window.

4. At this point, you can generate the formed command or you can choose to add some additional commands before generating.

CHAPTER 7

The Generate Screen - Working With MDL Files

MDL stands for MetaSuite Definition Language. This plain-text language is used to define the metadata (record and field structure) of Data Sources and Targets. MDL files containing the record and field definitions of one or more Data Sources or Targets are called Dictionary Files.

Note: If you are familiar with the MDL language, you can manually edit the MDL definitions. Refer to the chapter *Definition Language Commandes* in the MetaStore Manager Guide. However, this use is no longer recommended, as all Dictionary File editing can be performed from within the MetaSuite MetaStore Manager User Interface.

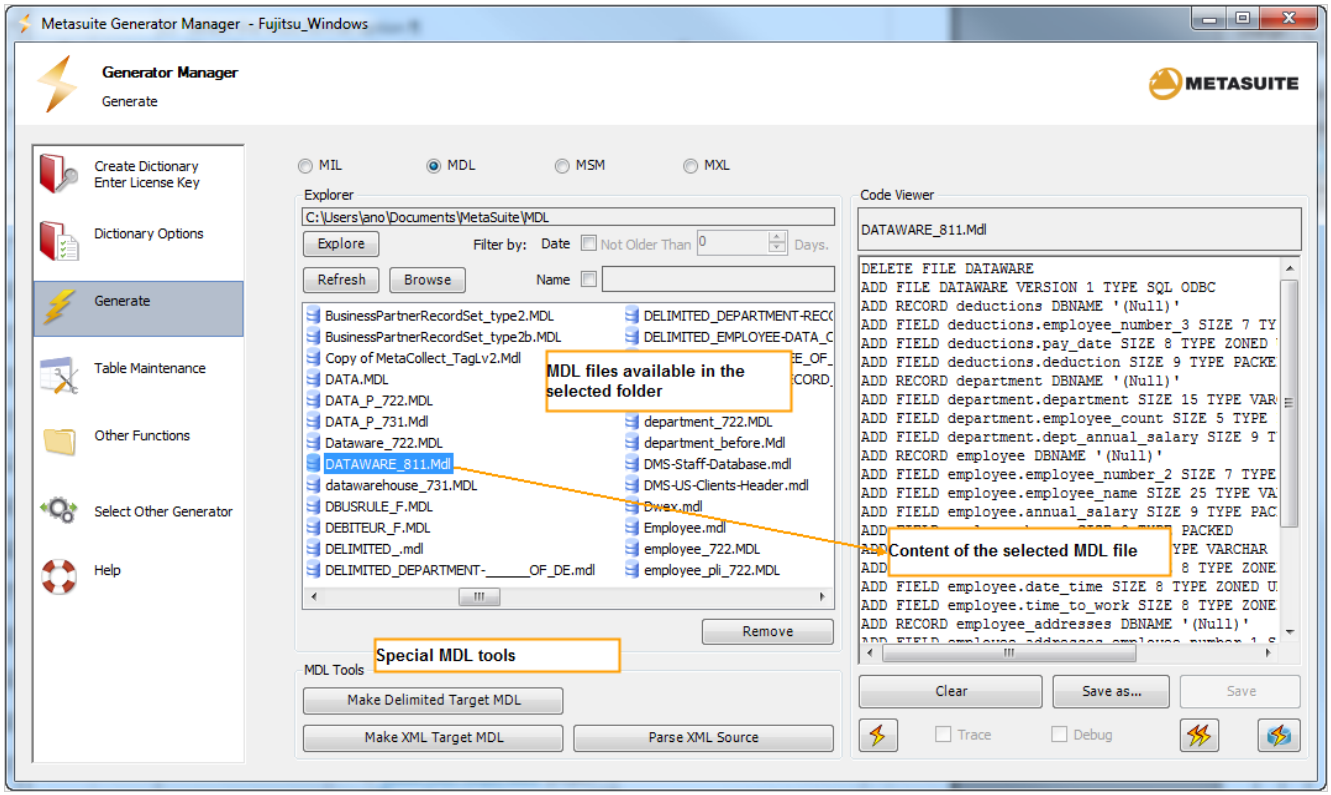
Refer to the following procedures for a detailed description:

- The Generate MDL Screen (page 37)
- Editing Existing MDL files (page 38)
- Using MDL Tools - Make Delimited Target MDL (page 39)
- Using MDL Tools - Make XML Target MDL (page 41)
- Using MDL Tools - Parse XML Source (page 44)

7.1. The Generate MDL Screen

1. Click the *Generate* icon in the left column.

2. Select the *MDL* format option.
   The following screen is displayed:



The MDL screen consists of three sections:

Section	Description
List of available MDL files	This list is displayed in the upper-left corner. Select a file to display its contents in the File content window on the right. See Editing Existing MDL files on page 38.
MDL Tools	Use the MDL Tools for special MDL transformations: • Make Delimited Target MDL (page 39) • Make XML Target MDL (page 41) • Parse XML Source (page 44)
MDL File content frame	This frame contains the MDL code contained in the selected MDL file or resulting from the selected MDL Tool operation. You can manually edit the displayed MDL code.

7.2. Editing Existing MDL files

1. Switch to the *Generate MDL* screen.
   See The Generate MDL Screen on page 37.

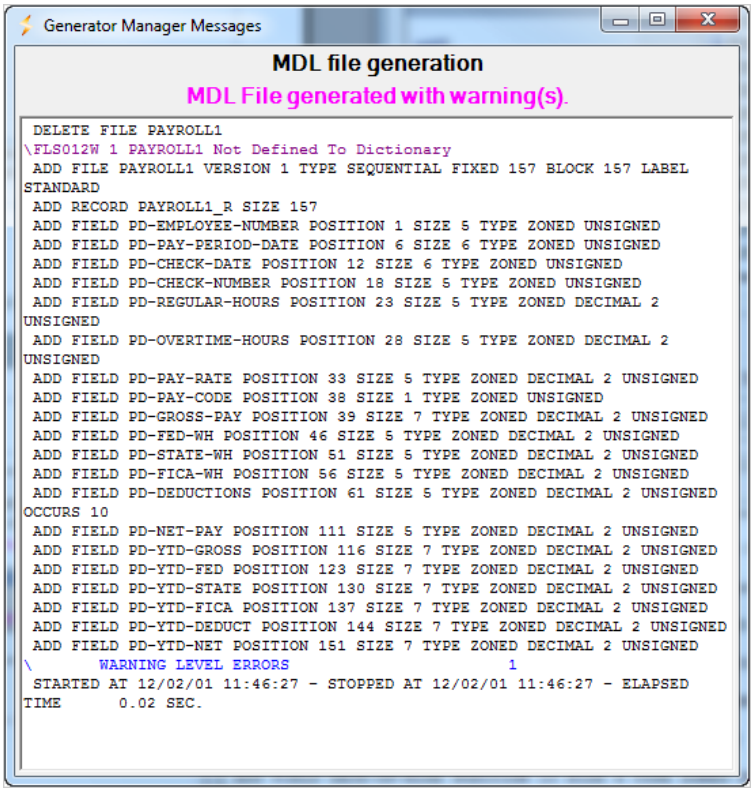
2. Select the required MDL file in the list of available MDL Files.
   Its content is immediately displayed in the frame to the right.

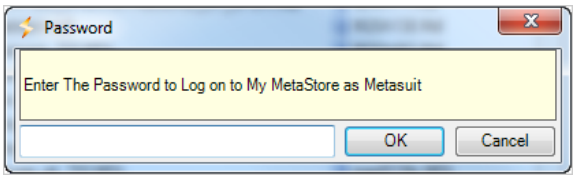
Note: If you select another MDL file now, the content of this file replaces the instructions already displayed. You can delete a MDL file, by selecting it in the list of available MDL files and then clicking the *Remove* button.

3. Edit the displayed MDL code, if required.

4. Once you have made the required changes, you can perform one of the following actions:

Action	Description
Clear	Clears the list content so that you can start over again.
Save as	Saves the displayed MDL code in a new MDL file. Enter the new Path and File name and click *OK*.
Save	Overwrite the existing MDL file with the displayed (edited) MDL code.
	Click the *Generate* button if you want to generate the MDL code. The MDL code will be generated, so that the new settings become valid in the MDL file. The following screen confirms this action:


Generator Manager Messages
MDL file generation
MDL File generated with warning(s).
DELETE FILE PAYROLL1
\FLS012W 1 PAYROLL1 Not Defined To Dictionary
ADD FILE PAYROLL1 VERSION 1 TYPE SEQUENTIAL FIXED 157 BLOCK 157 LABEL STANDARD
ADD RECORD PAYROLL1_R SIZE 157
ADD FIELD PD-EMPLOYEE-NUMBER POSITION 1 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD PD-PAY-PERIOD-DATE POSITION 6 SIZE 6 TYPE ZONED UNSIGNED
ADD FIELD PD-CHECK-DATE POSITION 12 SIZE 6 TYPE ZONED UNSIGNED
ADD FIELD PD-CHECK-NUMBER POSITION 18 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD PD-REGULAR-HOURS POSITION 23 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-OVERTIME-HOURS POSITION 28 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-PAY-RATE POSITION 33 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-PAY-CODE POSITION 38 SIZE 1 TYPE ZONED UNSIGNED
ADD FIELD PD-GROSS-PAY POSITION 39 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-FED-WH POSITION 46 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-STATE-WH POSITION 51 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-FICA-WH POSITION 56 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-DEDUCTIONS POSITION 61 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED OCCURS 10
ADD FIELD PD-NET-PAY POSITION 111 SIZE 5 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-GROSS POSITION 116 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-FED POSITION 123 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-STATE POSITION 130 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-FICA POSITION 137 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-DEDUCT POSITION 144 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PD-YTD-NET POSITION 151 SIZE 7 TYPE ZONED DECIMAL 2 UNSIGNED
\ WARNING LEVEL ERRORS 1
STARTED AT 12/02/01 11:46:27 - STOPPED AT 12/02/01 11:46:27 - ELAPSED TIME 0.02 SEC.

Action	Description
	Generate multiple MDL files at once. Select the files you want to generate and click the *Generate* button.
	Imports the MDL code files in Batch. The following screen is displayed:  Enter the password and click *OK*. The log file will be displayed.

7.3. Using MDL Tools - Make Delimited Target MDL

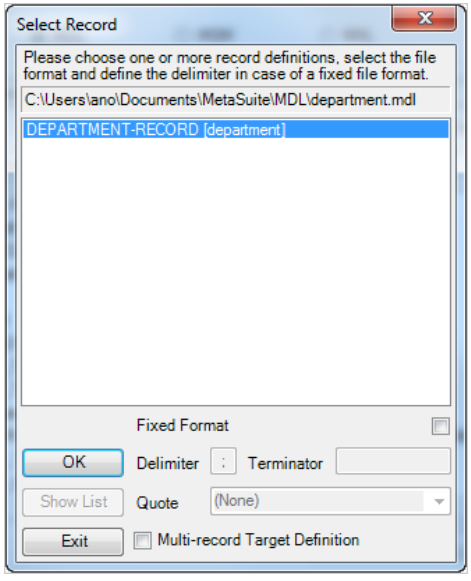
Delimited or comma-separated Target MDL Files are sequential files that contain field values, where each field value is separated by a semi-colon or by another character. Some delimited files also contain a row terminator character.

Because of their portability, they can be useful if you want to make conversions to another operating system.

1. Switch to the *MDL* screen.
   See The Generate MDL Screen on page 37.

2. From the available file list, select the MDL file you want to make a Delimited Target MDL file for.

3. Select the option *Make Delimited Target MDL* from the MDL Tools section.
   The following window is displayed:



4. At the top of the window, select one or more Records that should appear in the Delimited Target MDL file.

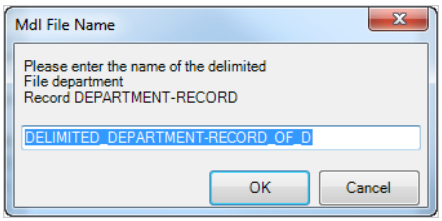
5. Below the Record list, set the options as required.

The following options are available:

Option	Description
Fixed Format	Select this check box, if you require a fixed format. Each field will have a predefined starting point. Clear this check box, if you require variable output format. The fields will be placed one behind the other without spaces between them.
Delimiter	In this field, overwrite the default Delimiter character, if required. The default character is a semi-colon.
Terminator	In this field, overwrite the default Terminator character, if required. The default value is an empty string.
Quote	In this field, overwrite the default Quote setting, if required. You can surround alphanumeric strings with quotes. This is useful if spaces and the delimiter or row terminator characters can be part of the string and should not be recognized as such.
Multi-record target definition	Select this check box, if you want to create one MDL file definition containing all record definitions. The target file will also be a multi-record file. Clear this check box, if you want to create one file definition per record definition. Each target file will contain a single record definition.

6. Once you have made all required selections, click *OK*.

The following window is displayed:



The suggested default name is constructed as follows:

DELIMITED[_RECORDNAME]_FILENAME

Note:

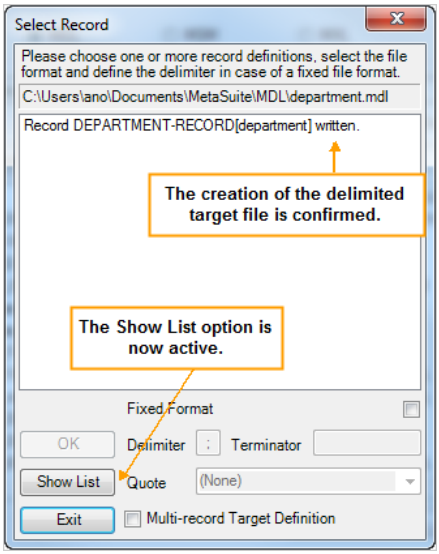
- The RECORDNAME indication is not available for multi-record target files.
- The maximum file name length is 32 characters. Longer names are truncated.
- The MDL extension is added automatically.
- The generated Delimited Target MDL File is saved in the default MDL folder, as shown in the path at the top of the screen.

7. Change the default name if required, and click *OK*.

Results:

- The Delimited Target MDL file will be generated and shown in the list of available MDL files.

- The *Select Record* window is redisplayed:



8. Click *Show List* to display a report in the following format:

department

Previous Record Next Record

N°	Name	Size	Type	Description
1	FIELD_1	103	character	-

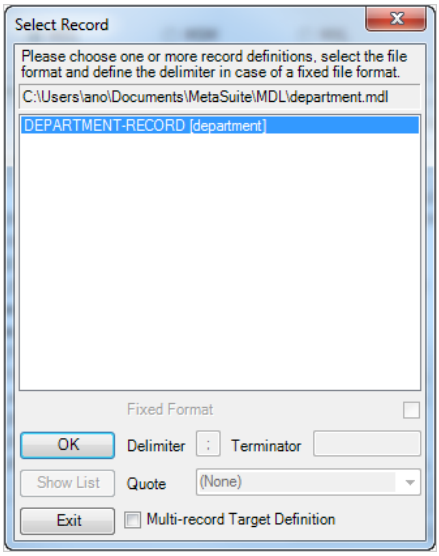
7.4. Using MDL Tools - Make XML Target MDL

XML Target MDL Files are sequential target files with XML content.

1. Switch to the *MDL* screen.
   See The Generate MDL Screen on page 37.

2. From the available file list, select the MDL file you want to make an XML Target MDL file for.

3. Select the option *Make XML Target MDL* from the MDL Tools section.
The following screen is displayed:

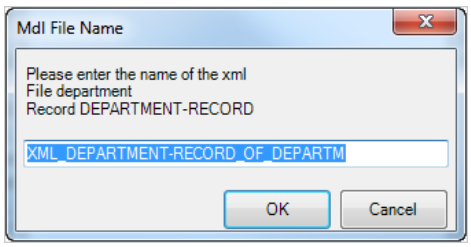


4. At the top of the window, select one or more Records that should appear in the XML Target MDL file.

5. Below the Record list, set the options as required.
The following options are available:

Option	Description
Fixed Format	This option is not available for XML Target MDL files.
Delimiter	This option is not available for XML Target MDL files.
Terminator	This option is not available for XML Target MDL files.
Quote	This option is not available for XML Target MDL files.
Multi-record target definition	Select this check box, if you want to make one MDL file definition containing all record definitions. Clear this check box, if you want to make a separate MDL file definition for each record definition.

6. Once you have made all required selections, click *OK*.
The following window is displayed:



The suggested default name is constructed as follows:
XML[_RECORDNAME_OF]_FILENAME

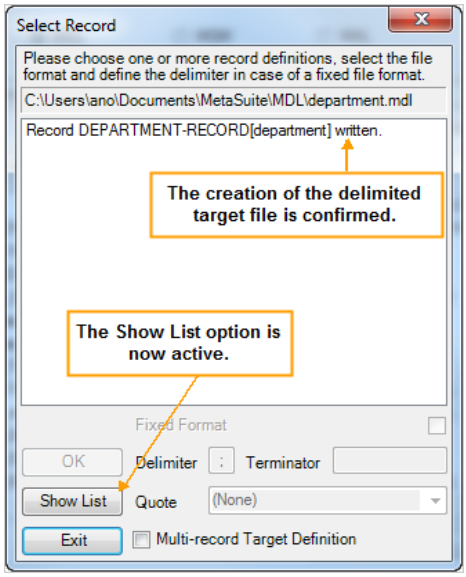
Note:

- The RECORDNAME_OF indication is not available for multi-record target files.
- The maximum file name length is 32 characters. Longer names are truncated.
- The MDL extension is added automatically.
- The generated Delimited Target MDL File is saved in the default MDL folder, as shown in the path at the top of the screen.

7. Change the default name if required, and click *OK*.

Results:

- The XML Target MDL file will be generated and shown in the list of available MDL files.
- The *Select Record* window is redisplayed:



8. Click *Show List* to display a report in the following format:

department

Previous Record Next Record

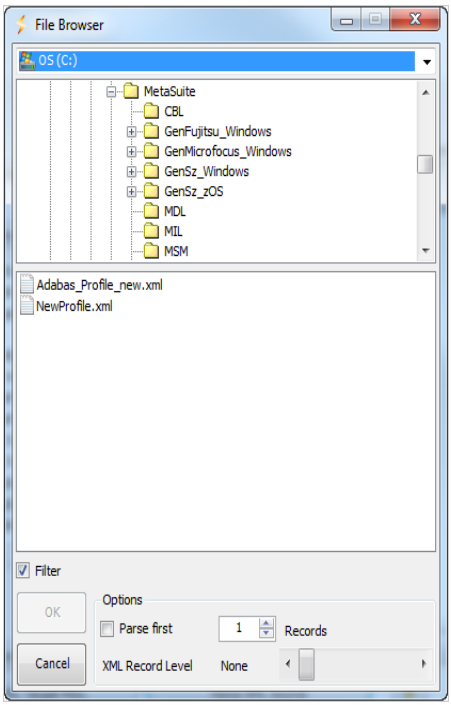
N°	Name	Size	Type	Description
1	FIELD_1	103	character	-

7.5. Using MDL Tools - Parse XML Source

XML source files can be parsed to MDL, so that they can be used within MetaSuite.

1. Switch to the *MDL* screen.
   See The Generate MDL Screen on page 37.

2. Select the option *Parse XML Source* from the MDL Tools section.
   The following screen is displayed:



3. Browse to the required folder and select the required File from the list.

Note: Select the *Filter* check box to only list XML files.

4. Below the file list, set the options as required.
   The following options are available:

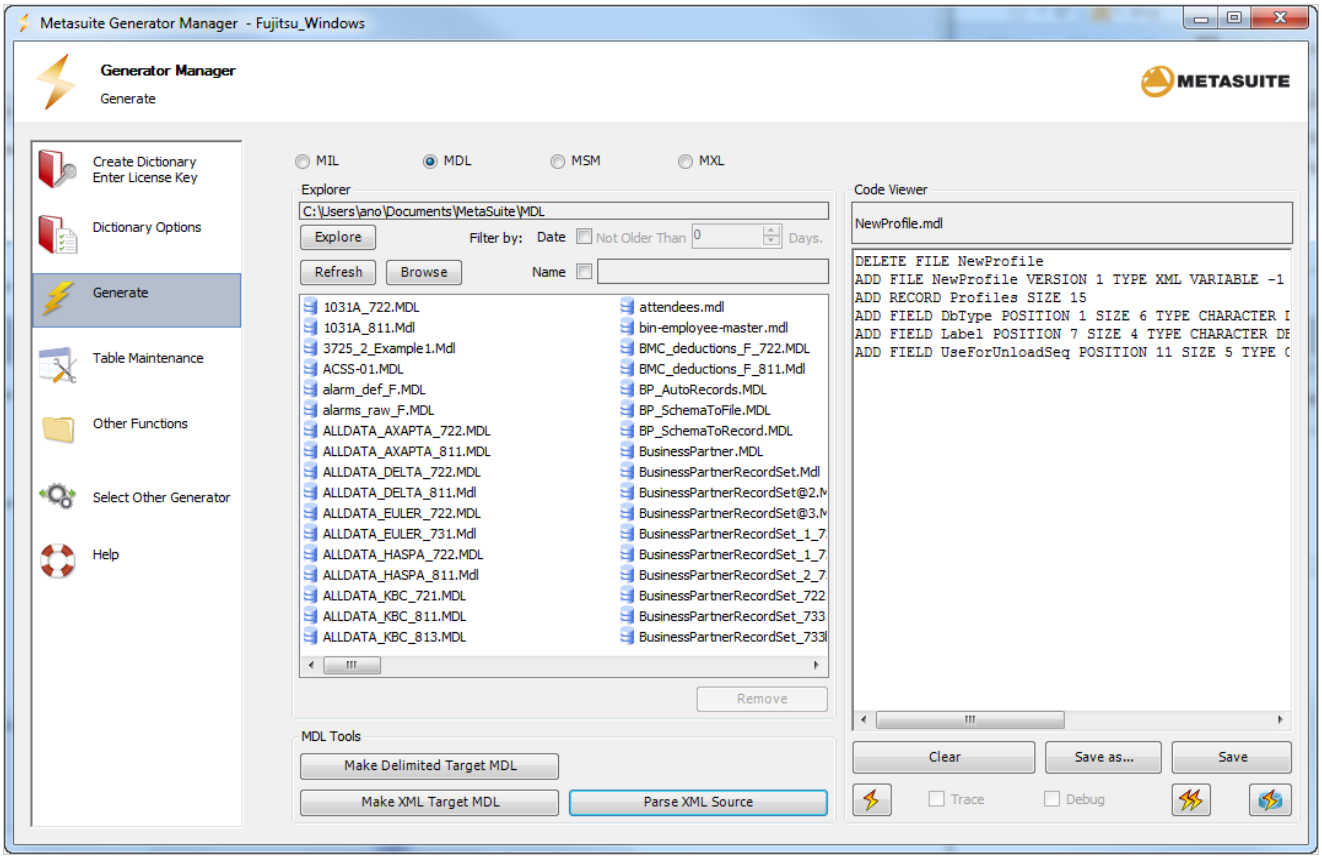
Option	Description
Parse first [number] records	Check this check box to parse only a number of records at the start of the file. If you check the check box, you should also provide the number of records to be parsed.
Delimiter	In this field, overwrite the default delimiter character, if required.

Option	Description
Row terminator	In this field, overwrite the default row terminator, if required.
XML record level	Select the required level: • Select *0*, if no XML levels must be taken into account. The XML fields will be parsed, regardless of their XML tag level. This means that all fields must have a unique name. • Select *1*, if all XML levels must be taken into account. The first XML Level will be selected as "record level". Tags on this level will contain the record name. • Select *2*, if XML levels 2 and higher must be taken into account. Tags on the first level will be skipped. Level 2 XML tags will be selected as "record level". • Selecting a higher value will result in a higher level being considered as the record level.

5. Once you have made all required selections, click *OK*.

   The XML Source file is parsed and the resulting MDL file is added to the list of available MDL files. The file name is maintained, only the extension is changed to MDL.

   Sample XML file:



6. Edit the MDL file as required.

CHAPTER 8

The Generate Screen - Working With MSM Files

MSM stands for MetaSuite Model files. These files contain the definitions of the Data Sources, Data Targets and the mapping rules between them *in a coded form*.

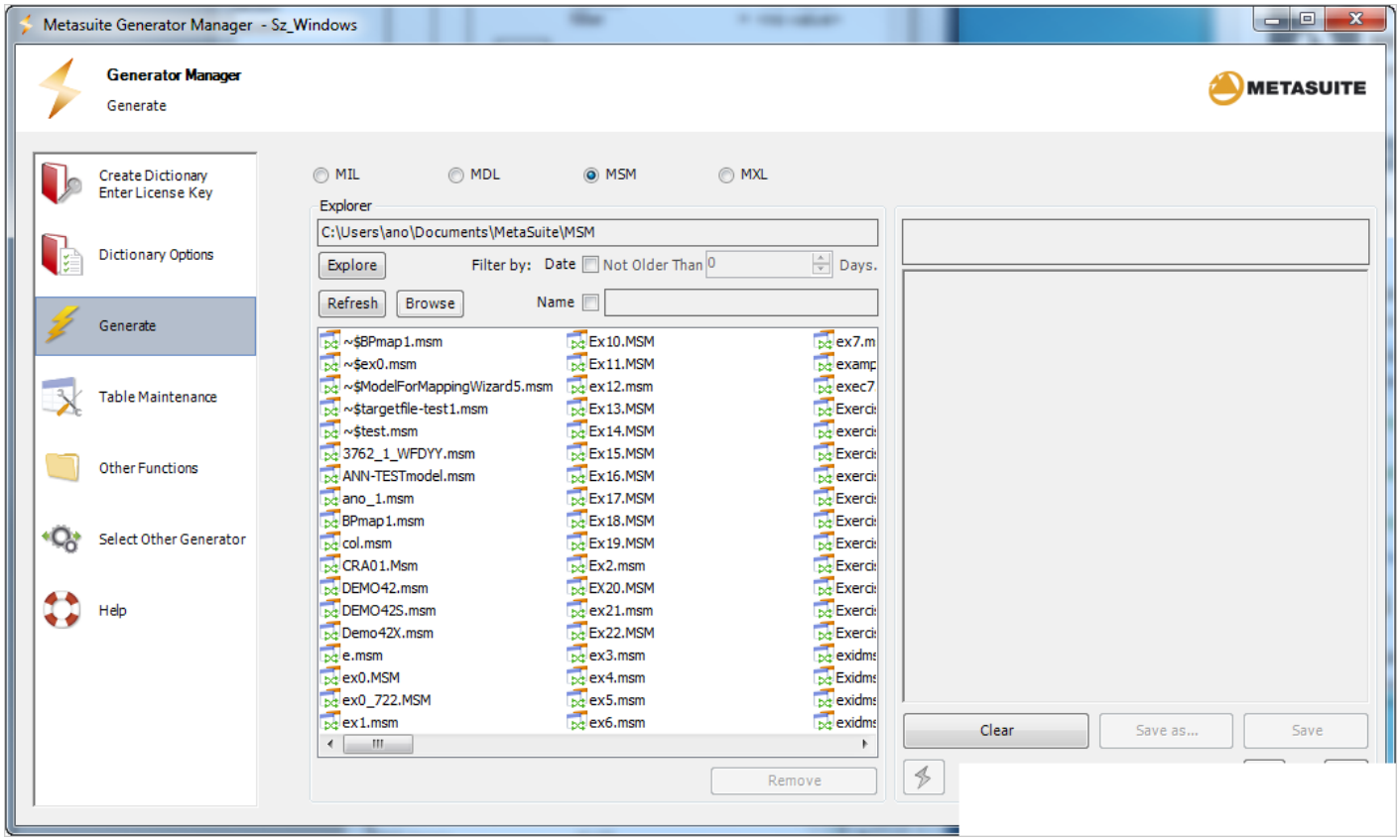
MetaSuite Models can be edited by means of the MetaMap Manager application.

The MetaSuite Model also exists in a plain-text version, called MetaSuite Extraction Language (MXL). See The Generate Screen - Working With MXL Files on page 48.

MSM files can be exported in batch. Follow the procedure described below.

1. Click the *Generate* icon in the left column.

2. Select the *MSM* format option.
   The following screen is displayed:



The available Models are listed at the left.

3. Two options are available:

3.1. Generate Multiple in Batch

If you need to generate multiple MSM files at once, click the *Generate Multiple in Batch* button ( ).

Select the files you want to generate and click the *Generate* button.

3.2. Export MetaSuite Model in Batch

To export the MetaSuite model in batch, click the *Export MetaSuite Model in Batch* button ( ).

The matching MXL file will be generated.

CHAPTER 9

The Generate Screen - Working With MXL Files

MXL stands for MetaSuite eXtraction Language files. These files contain the definitions of the Data Sources, Data Targets and the mapping rules between them *in plain text*.

MXL files can be generated. See Generating an MXL file on page 49.

It is also possible to generate Multiple MXL files in batch. This is interesting in case a new generator build has been installed and you want to regenerate some or all MXL files. See Generating Multiple MXL Files in Batch on page 51.

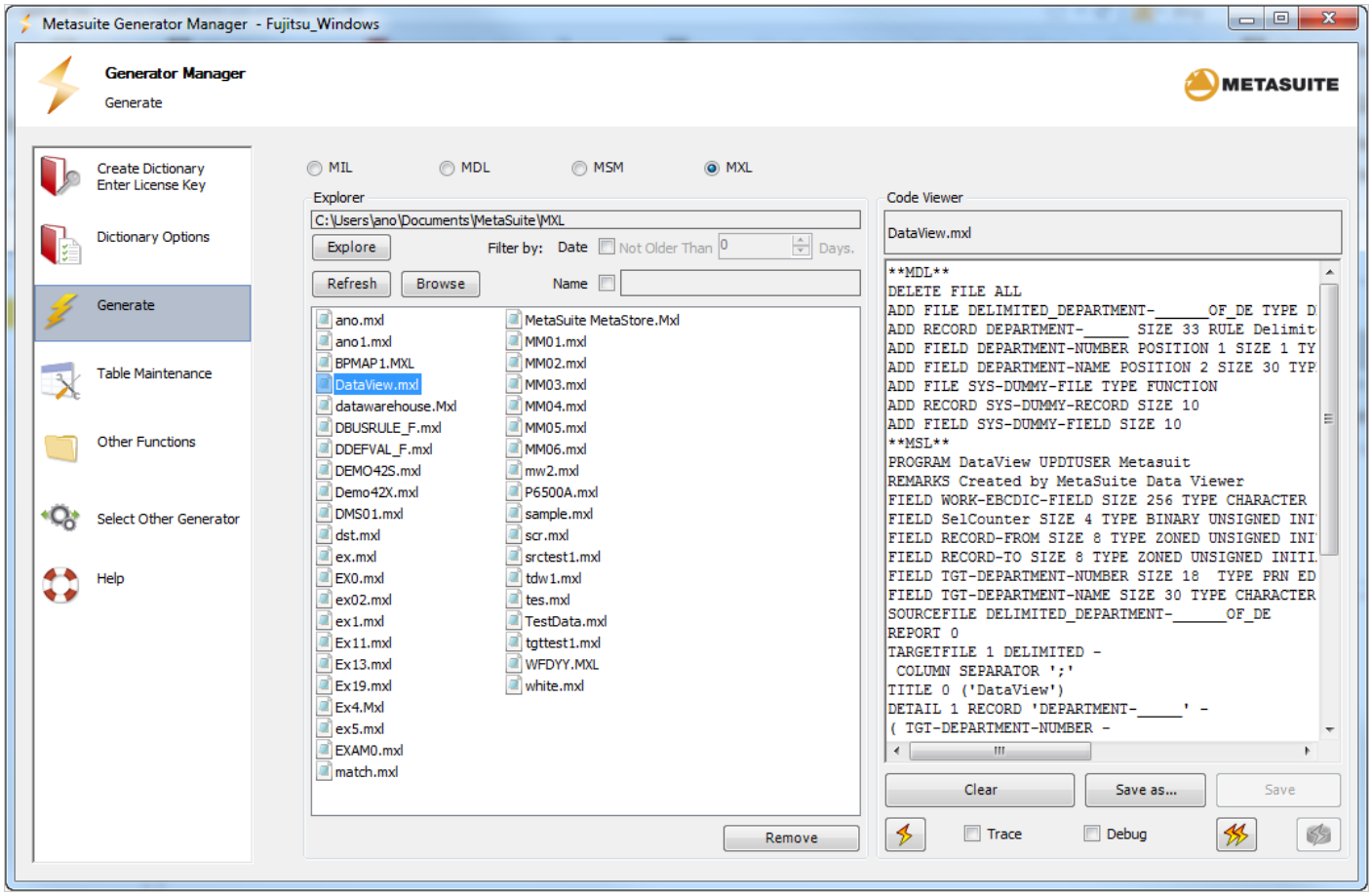
Refer to the following procedures for a detailed description:

- Generating an MXL file (page 49)
- Generating Multiple MXL Files in Batch (page 51)

9.1. Generating an MXL file

1. Click the *Generate* icon in the left column.

2. Select the *MXL* format option.
   The following screen is displayed:



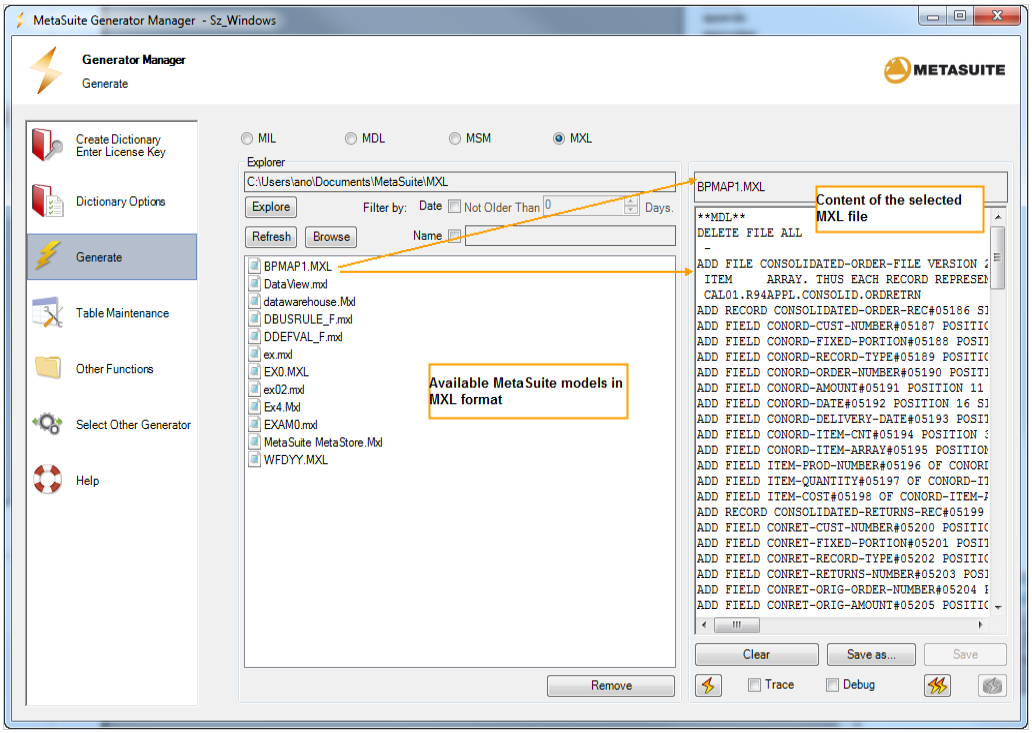
   The available MXL files are listed to the left.

3. Select the required MXL File.
   Its content is displayed in the frame on the right.

Note: If you are familiar with the MXL language, you may directly edit the code. Refer to the chapter *Definition Language Commands* in the *MetaStore Manager User Guide* for more detailed information. However, you are advised to perform changes from within the MetaMap Manager.

4. Once you have made the required changes, you can perform one of the following actions:

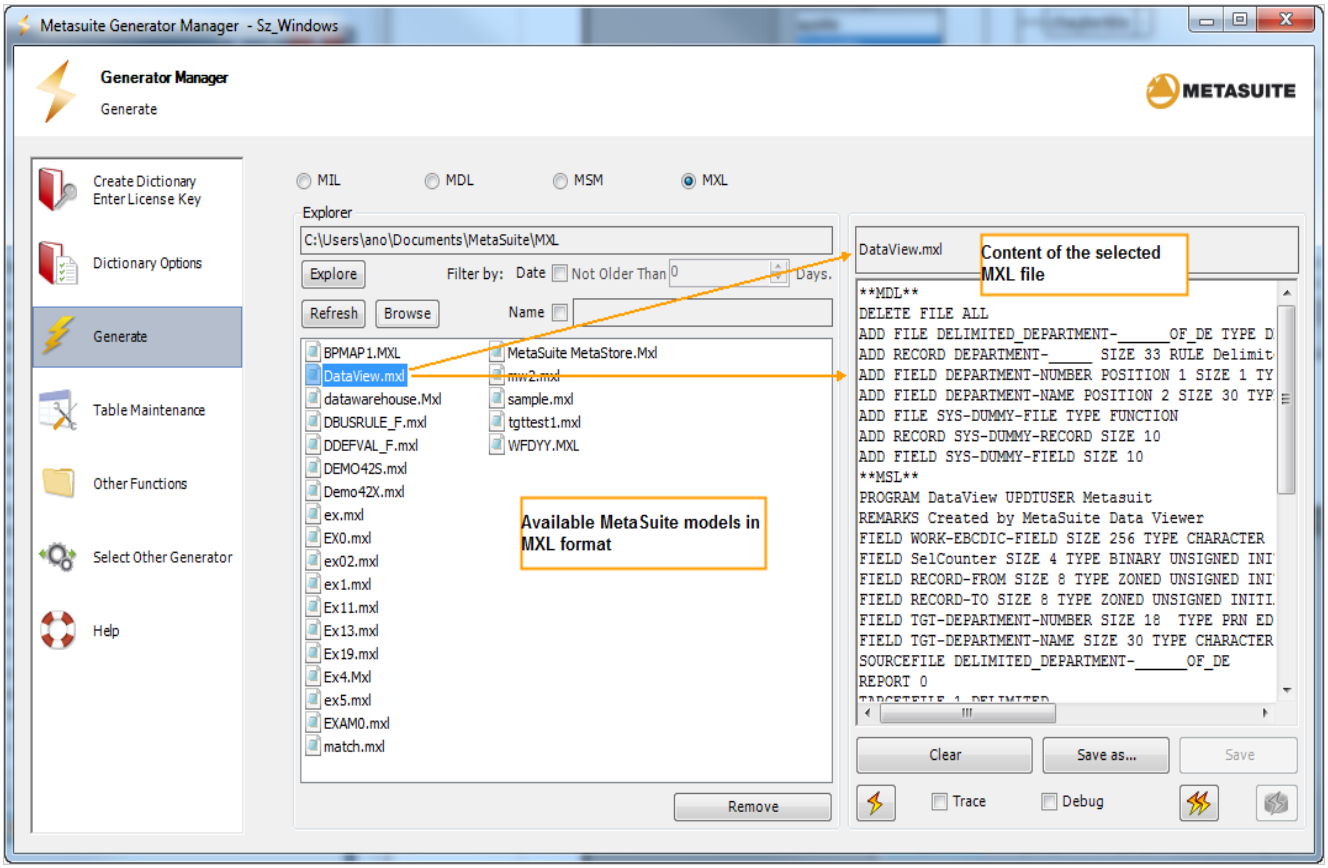
Action	Description
Clear	Removes all code so that you can start over again.
Save As...	Save the code in its current form in a new MXL file.
Save	Saves the code in its current form in the existing MXL file.

Action	Description
	Generates the MXL file. This is interesting to check the file for syntax errors. You can select the Trace and/or Debug check boxes to include additional information. Unless you specified different default folders in the INI Manager, the resulting MGL file (COBOL program) and MRL file will be saved in the MGL and MRL folders in the root directory of the Generator Manager being used. 
	Generates multiple MXL files at once. Select the files you want to generate and click the *Generate* button.

9.2. Generating Multiple MXL Files in Batch

1. Click the *Generate* icon in the left column.

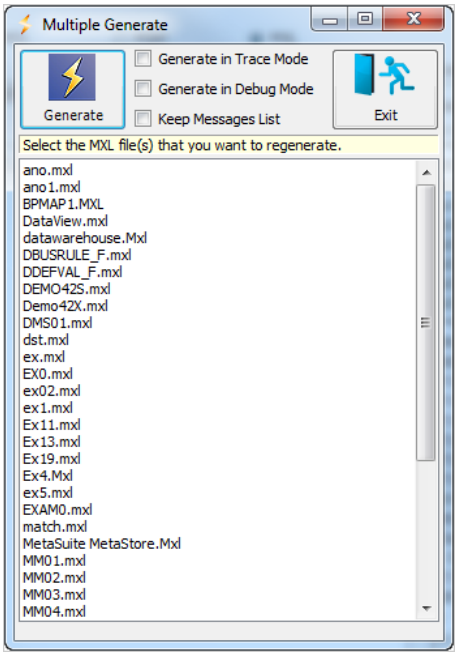
2. Select the *MXL* format option.
   The following screen is displayed:



The available Models are listed at the left.

3. Click the button.

   The following dialog is displayed:



4. Select the files you want to generate in batch and click *Generate*.

   You can select the Trace and/or Debug check boxes to include additional information.

   Unless you specified different default folders in the INI Manager, the resulting MGL file (COBOL program) and MRL file will be saved in the MGL and MRL folders in the root directory of the Generator Manager being used.

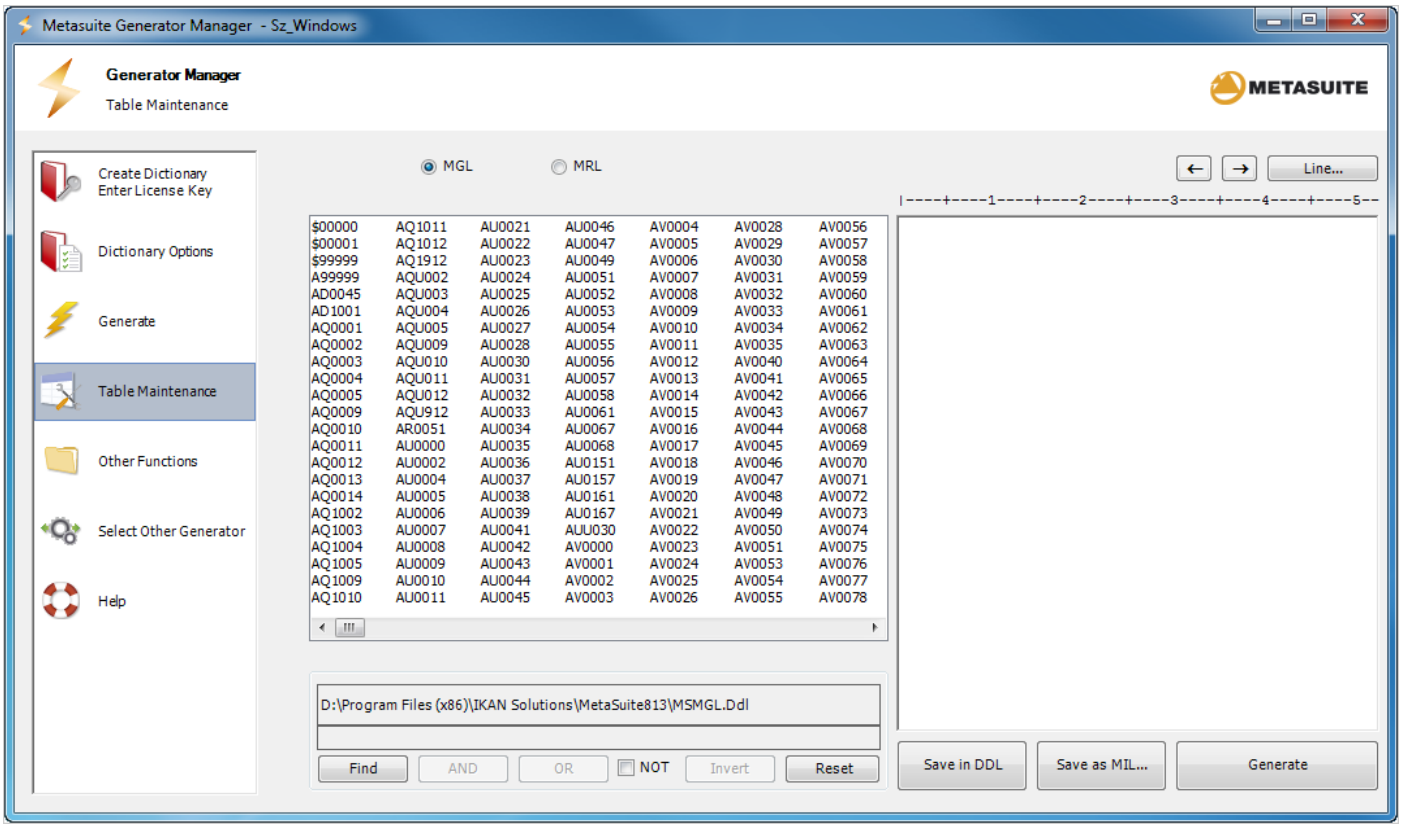
CHAPTER 10

The Table Maintenance Screen - Overview

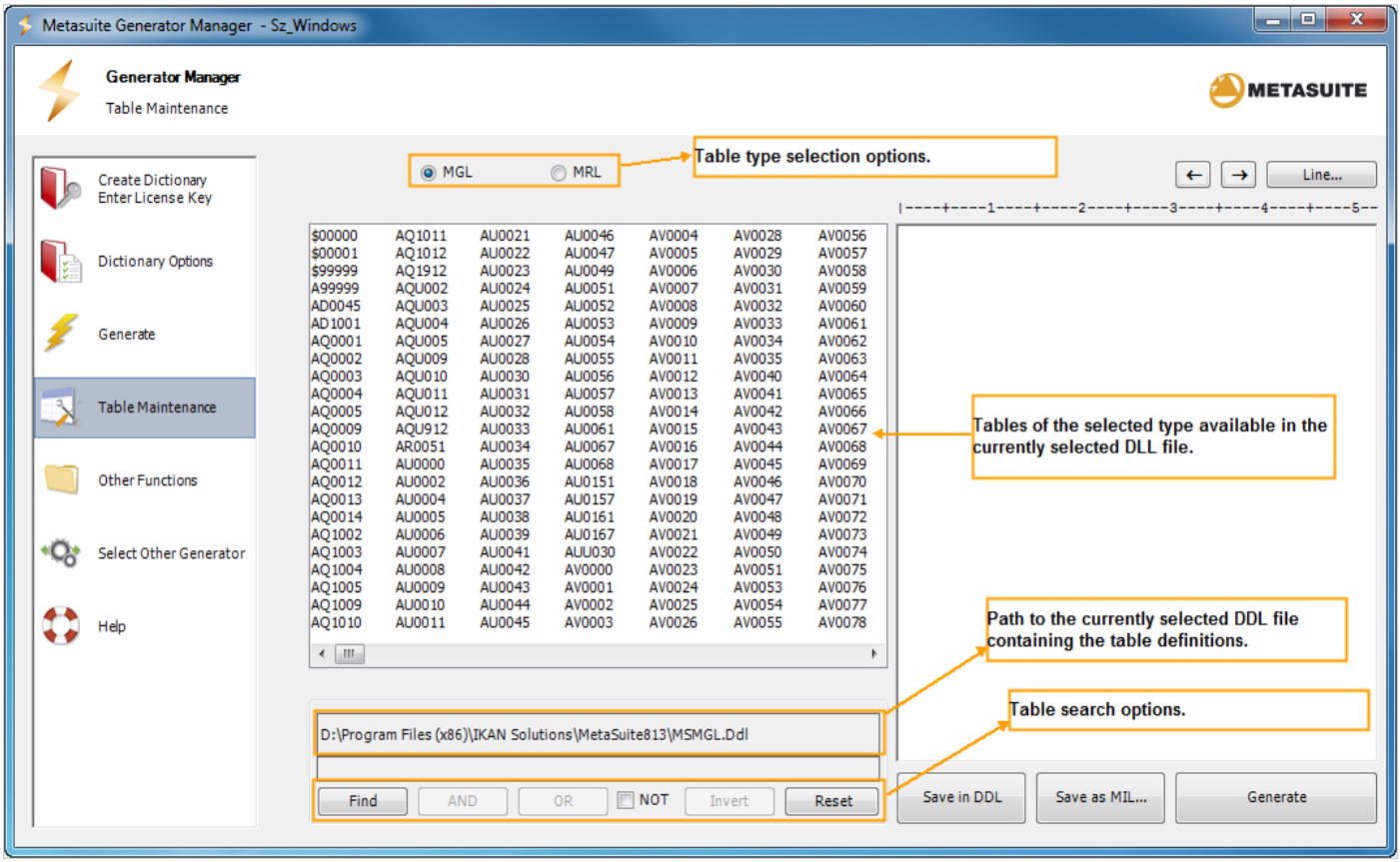
The *Table Maintenance* screen allows you to generate the different file types: MGL and MRL.
Refer to the dedicated chapters for more specific information:

- The Table Maintenance Screen - Managing MGL Tables (page 57)
- The Table Maintenance Screen - Managing MRL Tables (page 75)

1. Switch to the *Table Maintenance* screen by clicking the *Table Maintenance* icon in the left column. The following screen is displayed:



The common elements available are listed on the left:



Note: The other elements are specific for each file type. Refer to the dedicated chapters.

2. Select the required file type.

   The following types are available:

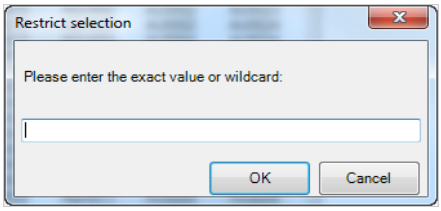
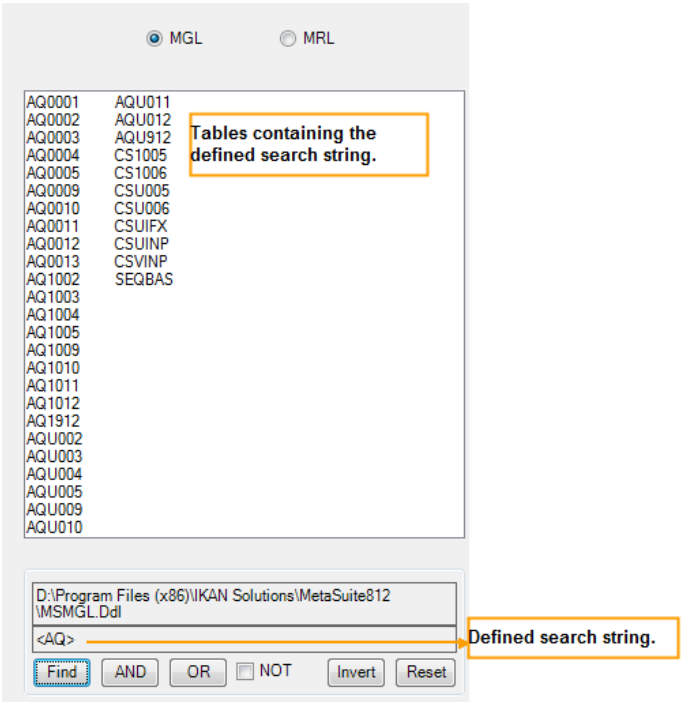
   - MGL (page 57)
   - MRL (page 75)

3. Verify the path and file name displayed below the list of available tables.

   It refers to the DLL file containing the definition of the displayed tables.

4. If you do not immediately find the table you require in the list of available tables, you can use the Table Search options.

   These options allow checking if defined character strings are available in the Tables *content*. However, as each table contains its name on the first line, you can also use them to find specific table names.

Option	Description
Find	Click *Find* to define a search string. The following screen is displayed:  Define the search string and click *OK*. Result: • The defined search string is displayed above the search options. • The list of displayed tables is limited to tables containing the search string. 
AND	This option allows combining two search strings. Tables will only be displayed, if they contain BOTH search strings. • Use the *Find* option to define the first search string. • Click the *AND* option and enter the second search string. Result: • The defined search string is displayed above the search options. • The list of displayed tables is limited to tables containing both search strings.
OR	This option allows combining two search strings. Tables will be displayed, if they contain one or both of search strings. • Use the *Find* option to define the first search string. • Click the *OR* option and enter the second search string. Result: • The defined search string is displayed above the search options. • The list of displayed tables is limited to tables containing one or both search strings.

Option	Description
NOT	This option allows negating a defined search string. Tables will only be displayed, if they do not contain the search string: • Select the *NOT* check box. • Click *Find* and enter the search string to be negated. • Click *OK*. Result: • The defined search string is displayed above the search options. • The list of displayed tables is limited to tables NOT containing the defined search string.
Invert	This option allows inverting a combined search string. Tables will be displayed, if they do NOT match the defined search string • Define the search string as usual, with the FIND, AND and OR options. • Click *Invert* to invert the complete search string Result: • The defined search string is displayed above the search options. • The list of displayed tables is limited to tables NOT matching the defined search string.
Reset	Click *Reset* to clear the defined selection criteria. All tables will be displayed again.

CHAPTER 11

The Table Maintenance Screen - Managing MGL Tables

MGL stands for MetaSuite Generate Language. The *MSMGL.DDL* file (located in the MetaSuite installation folder) is one of three upload files for the Generator Dictionary. The *MSMGL.DDL* file contains MGL Tables consisting of small pieces of COBOL code, required by the Generators to work properly.

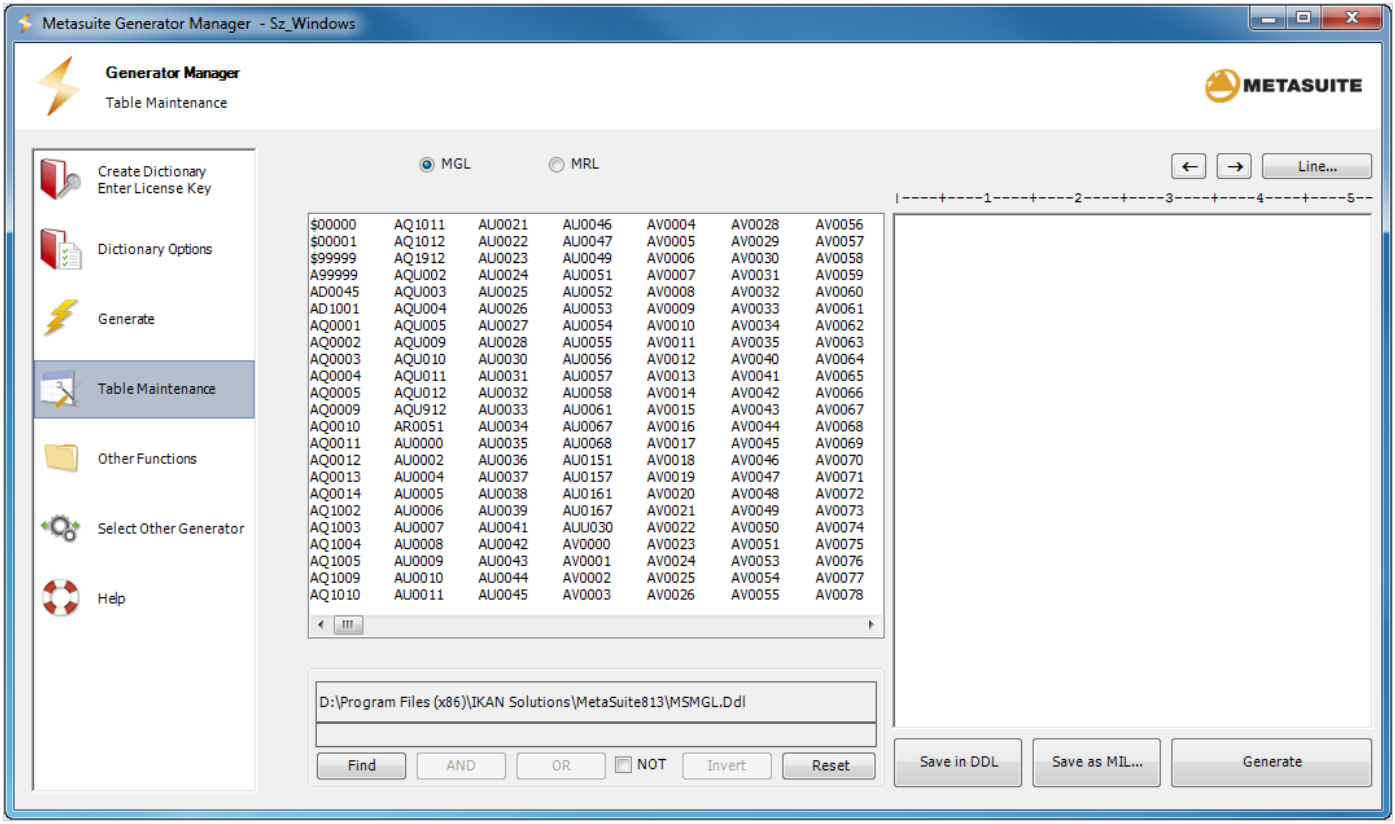
It is logical to create a new dictionary after editing the MGL Tables (experienced users only!).

Refer to the following procedures for a detailed description:

- The MGL Table Maintenance Screen (page 58)
- MGL Tables: Detailed Description (page 63)
- Code-control Tables (page 64)
- COBOL Code Tables (page 71)
- Code Table Examples (page 72)
- Creating Customized Code-control Tables (page 73)
- Creating Customized COBOL Code Tables (page 74)

11.1. The MGL Table Maintenance Screen

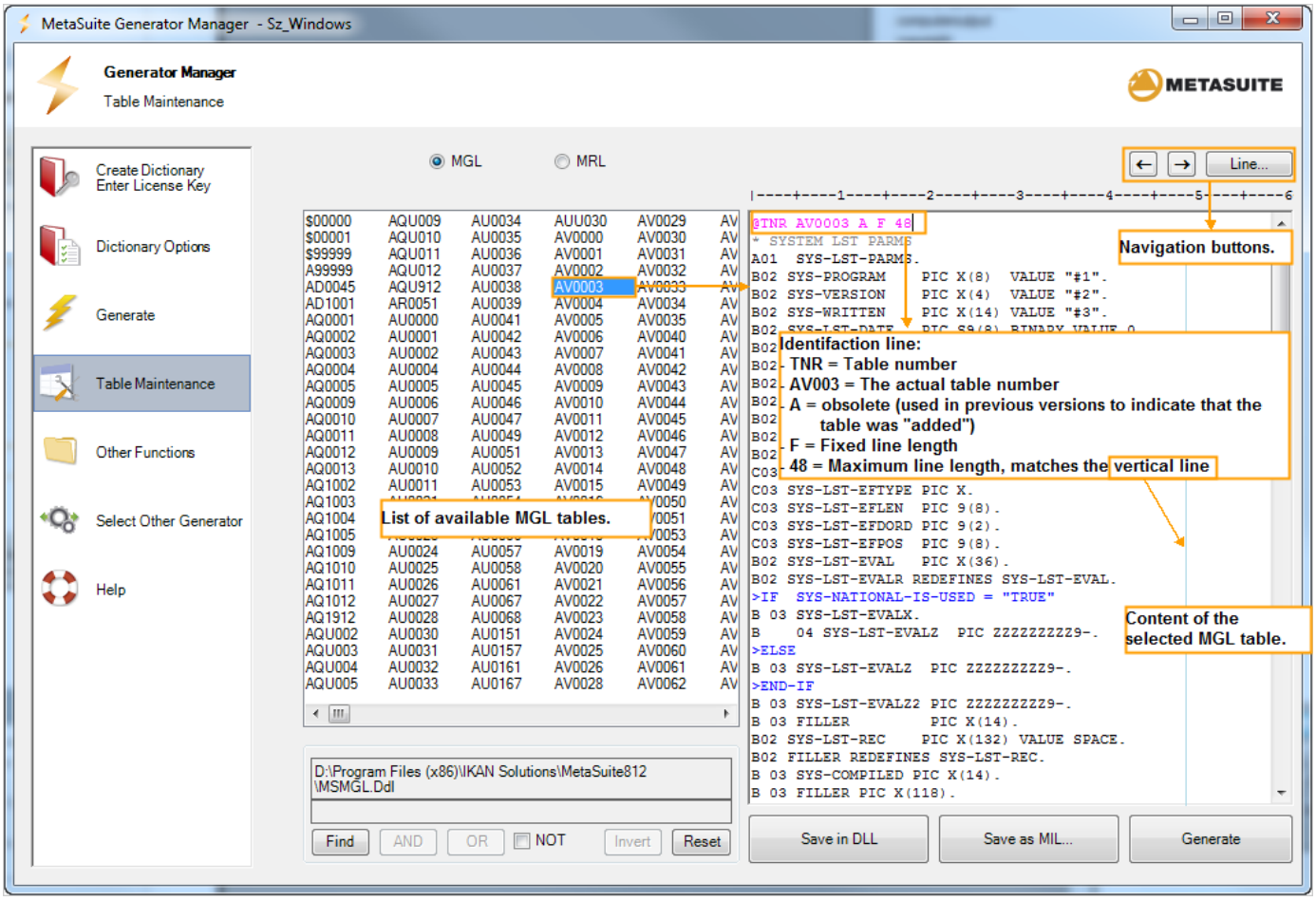
1. Click the *Table Maintenance* icon in the left column.
   The following screen is displayed:



2. Select the *MGL* format option.
   The available MGL tables are displayed.

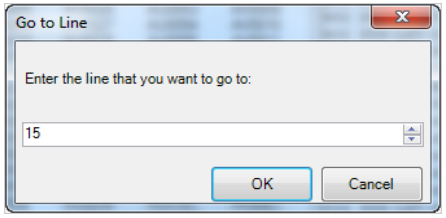
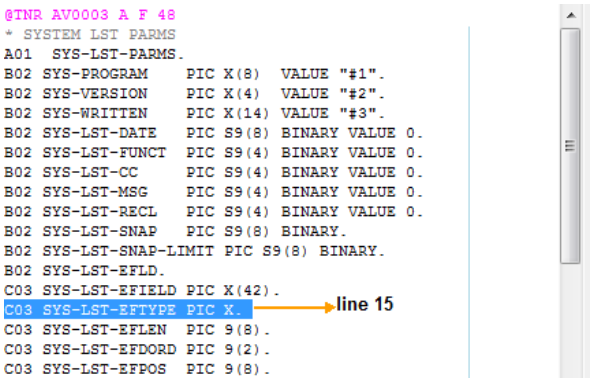
Note: If you want to limit the list of displayed MGL tables, you can define search strings as explained in chapter The Table Maintenance Screen - Overview (page 53).

3. Click an MGL table to display its contents on the right.
   The following screen is displayed:



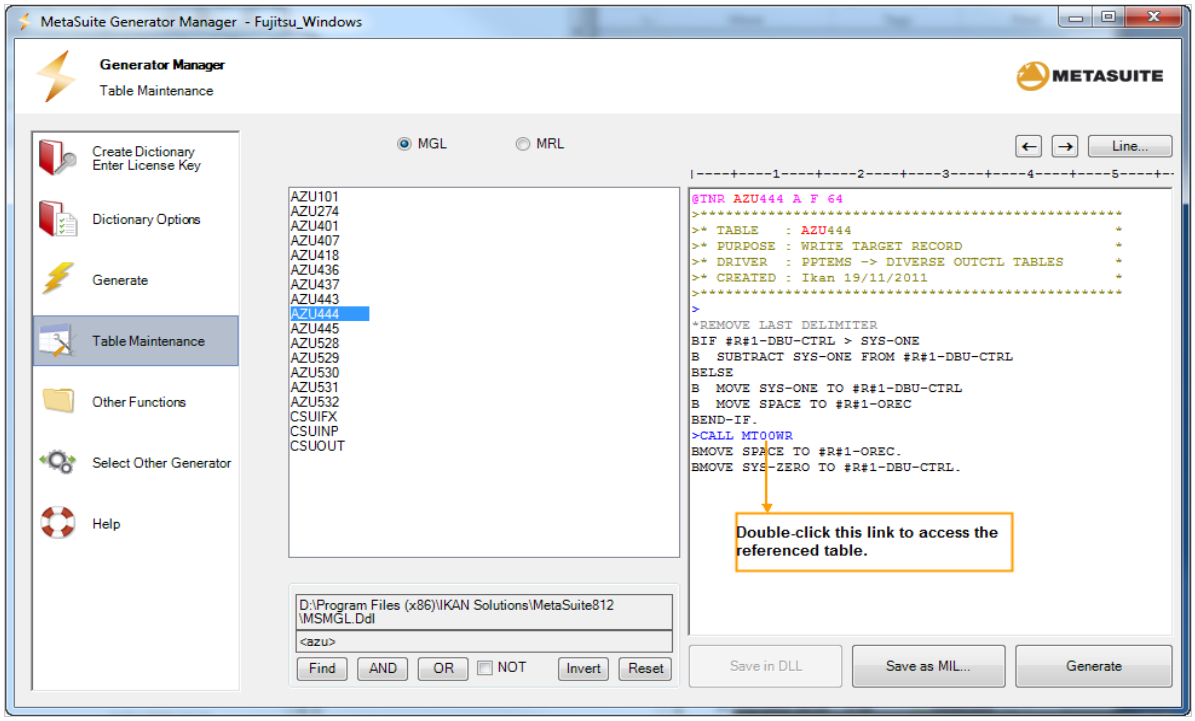
4. Use the Navigation buttons as required.
   The following buttons are available:

Icon	Option	Description
←	Previous table	If you have displayed a sequence of Tables, you can select this option to backward in the sequence and redisplay the previous MGL Table. By repeatedly clicking this option, you can return to the last 10 tables. If there is no more previous table to be shown, the option becomes inactive.

Icon	Option	Description
	Next table	If you have gone backward in the sequence of displayed tables by means of the *Previous table* option, you can select this option to go forward in the sequence of displayed MGL Tables. Once you have reached the last (most recent) table of the sequence, the option becomes inactive.
	Go to line number	Click this button to go to a specific line in the currently displayed Table. The following dialog is displayed:  Enter the required line number and click *OK*. Result: The selected line is highlighted in the file:  **Note**: Lines starting with an @ will not be counted.

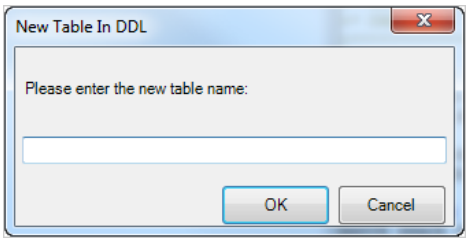
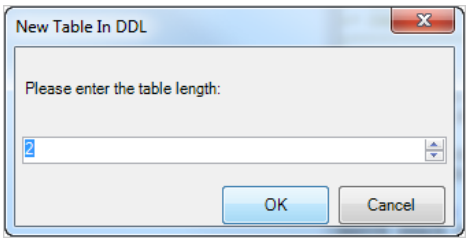
@INR AV0003 A F 48
* SYSTEM LST PARMS
A01 SYS-LST-PARMS.
B02 SYS-PROGRAM PIC X(8) VALUE "#1".
B02 SYS-VERSION PIC X(4) VALUE "#2".
B02 SYS-WRITTEN PIC X(14) VALUE "#3".
B02 SYS-LST-DATE PIC S9(8) BINARY VALUE 0.
B02 SYS-LST-FUNCT PIC S9(4) BINARY VALUE 0.
B02 SYS-LST-CC PIC S9(4) BINARY VALUE 0.
B02 SYS-LST-MSG PIC S9(4) BINARY VALUE 0.
B02 SYS-LST-RECL PIC S9(4) BINARY VALUE 0.
B02 SYS-LST-SNAP PIC S9(8) BINARY.
B02 SYS-LST-SNAP-LIMIT PIC S9(8) BINARY.
B02 SYS-LST-EFLD.
C03 SYS-LST-EFIELD PIC X(42).
C03 SYS-LST-EFTYPE PIC X.
line 15
C03 SYS-LST-EFLEN PIC 9(8).
C03 SYS-LST-EFDORD PIC 9(2).
C03 SYS-LST-EFPOS PIC 9(8).

5. If a table refers to another table, you can double-click the table name to access it directly:



6. Edit the code as required.

Note: Right-clicking in the table content's window, displays a pop-up menu with several commands, such as copy, paste, print, ...

Option	Description
New	Right-click the table content's window and select *New* from the pop-up menu. The following dialog is displayed:  Enter the new table name. It must contain exactly 6 characters. Then click *OK*. The following dialog is displayed:  Enter the number of expected lines and click *OK*. The Table is created. You can now enter the required code and save the table to the DDL file by means of the *Save to DDL* option.

7. Once you have performed the required changes, you can use one of the following options: *Save in DDL*, *Save as MIL* or *Generate*.

Note: These options are also available on the pop-up menu.

- Save in DDL

  Click *Save in DDL* to save the currently displayed version of a Table into the MSMGL.DDL file.

  You need to do this:

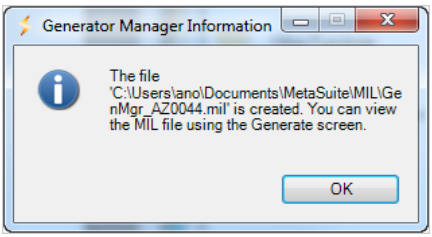
  - after editing an existing file
  - after creating a new file

  The new version of the table is saved in the `MSMGL.DDL` file.

- Save as MIL

  Click *Save as MIL* to save the changes to a table in a new MIL file in the standard MIL file directory.

  The following dialog is displayed:


Generator Manager Information
The file 'C:\Users\ano\Documents\MetaSuite\MIL\GenMgr_AZ0044.mil' is created. You can view the MIL file using the Generate screen.
OK

  The file name format is:

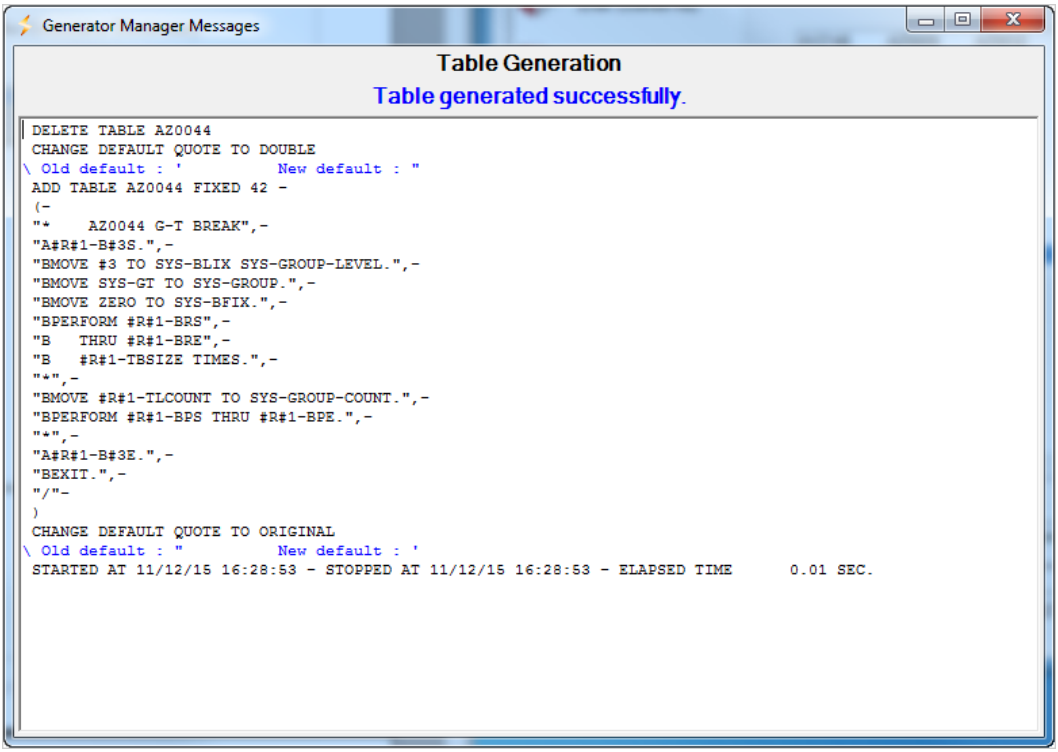
  GenMgr_TableName.MIL

  Implement the MIL file to make the changes effective. See The Generate MIL Screen on page 19.

- Generate

  Click *Generate* to make the changes immediately effective in the Generator Dictionary.

The following dialog is displayed:



11.2. MGL Tables: Detailed Description

When you have finished the creation of a MetaMap Model, you need to generate the matching COBOL source code program (MGL). For the generation of COBOL statements related to file processing, the Generator uses MGL tables.

There are 4 MGL table types:

- System Tables
- File Control Tables
- Code-control Tables
- COBOL Code tables

The following table describes those types:

MGL Table Type	Description
System Tables	System Tables are used by the generator for different purposes. Examples: • **AU0001**: contains all generator messages • **AU0002**: contains all reserved words • **AU0006**: contains the names of the system variables • **MTLOPT**: contains the MTL options • **MTLINI**: contains initial settings for some MTL variables.
File Control Tables	File Control Tables are still used by some data types instead of code-control Tables. They contain a single line consisting of 32 characters. Since their use becomes obsolete, they are not described in detail. Example: AU0030, AU0031

MGL Table Type	Description
Code-control Tables	Code-control Tables are available for each type of file supported by the system. Each of these code-control Tables comprises a fixed number of numbered entries that relate to a specific point in the COBOL program generation process. For example, there is an entry that determines what code is generated to read a file. We strongly recommend you NOT to edit the default code-control Tables. If a particular file or database requires special settings, it is better to create a new code-control Table specifically for this file or database. There are two code-control table categories: • Source code-control Tables (also called INPUT or GEN code-control Tables). See Source Code-control Tables on page 66. • Target code-control Tables (also called OUTPUT code-control Tables) See Target Code-control Tables on page 68.
COBOL Code Tables	Generator COBOL code tables are named collections of skeletal COBOL code. Each code table can have from one to many lines of COBOL code. Code tables are saved in the Generator Library (i.e. the MGL folder). It is possible to define ers within the COBOL code. They are signalled by a '#' followed by a single number or letter. A general convention used in the system causes file fields to be formatted in a code table as F#1-xxxx. During the actual code-generation process, the '#1' is replaced by the relative number of the file. See COBOL Code Tables on page 71.

11.3. Code-control Tables

The MetaSuite Generator uses code-control tables to control the generation of COBOL statements that relate to file processing in a MetaSuite generated COBOL program.

A code-control table is delivered with MetaSuite for each type of file supported by the system. Each of these code-control tables comprises a set of up to 54 numbered entries that relate to a specific point in the COBOL program generation process. For example, there is an entry that determines what code is generated to read a file.

We have two types of code-control tables:

1. Source code-control tables, containing up to 54 numbered entries, also called GEN or INPUT control tables.
2. Target code-control tables, containing up to 32 numbered entries, also called OUTPUT control tables.

During the MetaSuite Generator dictionary installation process, these standard code-control tables are stored. Each time a user ADDs a file definition to the dictionary, without specifying a code-control table in the ADD FILE command, the system defaults to the installed standard code-control table for the file type specified in the definition. A standard code-control table assumes that:

- The file can be opened, read, processed and closed using standard COBOL verbs.
- The file record(s) can be described using standard COBOL data definition statements.

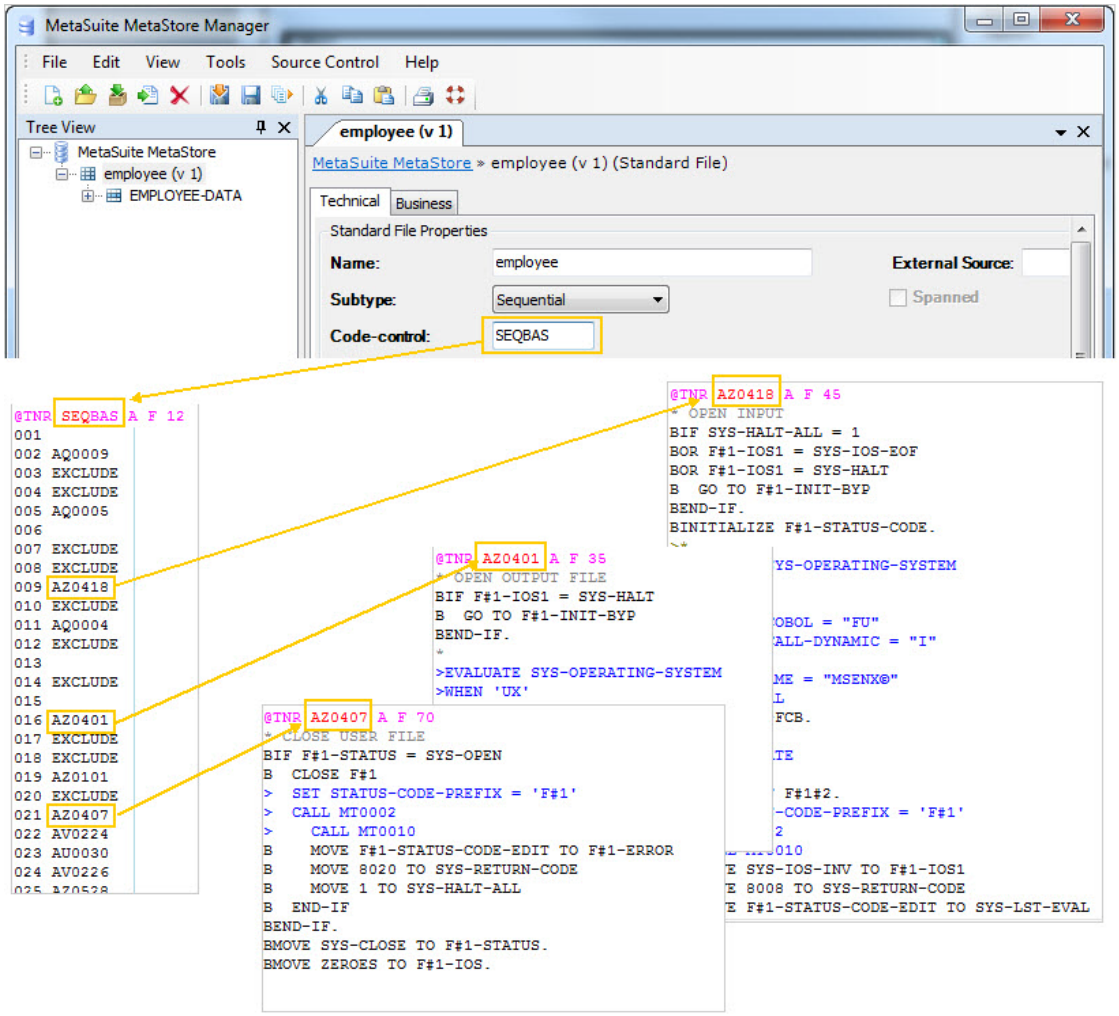
For the majority of files, the standard code-control tables provide all the information needed to generate COBOL code to read and process the files.

There are files, however, for which the standard code-control tables will not work because they do not meet the assumptions stated above (for example, a compressed file). In general there are two types of such files:

1. A file that cannot be accessed using standard COBOL verbs. A user-written subroutine, or set of subroutines, must be called to perform each I/O and any necessary decompression or reformatting.

2. A file that can be opened, read and closed using standard COBOL verbs, but whose records must be decompressed, or otherwise reformatted, by a user-provided subroutine immediately after being read and before any processing of the record data occurs.

The following figure shows how code-control tables are organized.



Format

Each standard source control table is a fixed-length table with 54 entries. Each entry is numbered, from 1 to 53, and is formatted as follows:

nnn xxxxxx

Where nnn is a 3-digit number (including leading zeroes), from 1 to 54.

xxxxxx is one of the following types of specification:

- Name of a COBOL code table, stored in the Generator dictionary, that is to be generated for the file at the related point in the COBOL generation process.
- EXCLUDE tells the MetaSuite Generator that no code table is to be generated for the file at the related processing point.
- Blanks tell the MetaSuite Generator to follow default code table generation.

  Note: This specification only applies to standard code-control tables. You cannot specify blanks for a code-control table entry in a user-written table.

Source Code-control Tables

The MetaSuite Generator Source code-control Tables are code-control tables used for treating source files.

Code-control tables for source files are defined in conjunction with the CODE-CONTROL option of the ADD FILE command, described in the chapter *Definition Language Commands* in the *Metastore Manager User Guide*.

The default source file control tables are:

AU0000	DUMMY file
AU0010	SEQUENTIAL flat file (IBM QSAM)
AU0023	INDEX file
AU0024	RANDOM file
AU0025	IBM VSAM KSDS file
AU0026	IBM VSAM ESDS file
AU0027	IMS file
AU0028	IDMS file
AU0034	TOTAL file
AU0035	ADABAS file
AU0039	DATACOM/DB
AU0046	DBS file
D20050	DB2 WINDOWS
D2I050	DB2 WINDOWS RESTART
D40050	DB2 AS4
D60050	DB2 RS6000
DM0050	DB2 Z/OS
DMI050	DB2 Z/OS RESTART
DV0050	DB2 DOS/VSE
FC0050	ODBC WINDOWS Fujitsu
IF0050	INFORMIX
IG0050	INGRES
IM0010	GSAM file
MQSINP	MQ-SERIES INPUT FILE
OR0050	ORACLE
PARINP	Parameter file input
RD0050	RDBMS
SE0050	SESAM (OSD/BS2000)
SEQBAS	Delimited (comma separated) flat file

SQ0050	SQLSERVER
SY0050	SYBASE
TD0050	TERADATA
XMLINP	XML (primitive - with no path control)
XPCIFX	XML with path control, fixed data format

The following table describes the entries in a standard Source code-control Table:

Entry	Function	Generates
001	SELECT	SELECT/ASSIGN statements
002	FD	FD statements
003	FDRECS	FD record layout
004	RECORD-PREP	One-time record preparation code
005	WS-MISC	Miscellaneous Working-Storage area
006	WS-RECS	Working-Storage record layout
007	LS-MISC	Miscellaneous Linkage-Section area
008	LS-RECS	Linkage-Section record layout
009	OPEN-INPUT	Open file code
010	POST-OPEN-IN	Post open input code
011	READ	Sequential read code
012	READ-KEYED	Keyed read code
013	POST-READ	Post read code
014	SET-SYSREC	Set SYSREC code
015	TASK-INIT	Start task code (after program init)
016	OPEN-OUTPUT	Open file output code
017	POST-OPEN-OUT	Post open output code
018	PRE-WRITE	Pre-write record code
019	WRITE	Sequential write code
020	WRITE-KEYED	Keyed write code
021	CLOSE	Close File code
022	BUF-TABHEAD	Buffer control code (Internal use)
023	FILE-CONTROL	File Control Table
024	BUF-TABEND	Buffer control code (Internal use)
025	BUF-READCTRL	Buffer control code (Internal use)
026	READ-IX-KEYED	Index key read code

Entry	Function	Generates
027	BUF-SIMPLOAD	Buffer control code (Internal use)
028	BUF-OCCLOAD	Buffer control code (Internal use)
029	BUF-LOADED	Buffer control code (Internal use)
030	BUF-EOF	Buffer control code (Internal use)
031	ENTRYNAME	Entry point name
032	ENTRYPARM	Entry point parameters
033	GETNAM	Record setup for keyed access
034	GETKEY	Key setup for keyed access
035	BUF-TABENT1	Buffer control code (Internal use)
036	BUF-TABENT2	Buffer control code (Internal use)
037	BUF-TABENT3	Buffer control code (Internal use)
038	BUF-TABENT4	Buffer control code (Internal use)
039	BUF-TABENT5	Buffer control code (Internal use)
040	BUF-TABENT6	Buffer control code (Internal use)
041	START	Start file code
042	FLIST-NAME	Formatted list code (Internal use)
043	FLIST-NAME1	Formatted list code (Internal use)
044	FLIST-HEADER	Formatted list code (Internal use)
045	FLIST-END	Formatted list code (Internal use)
046	TASK-START	Start task code (after program init)
047	TASK-STOP	Task stop code
048	DUPL-READ	Duplicate key read code
049	FD-DUMMY-REC	Skeletal FD record layout
050	WS-DUMMY-REC	Skeletal Working-Storage record
051	LS-DUMMY-REC	Skeletal Linkage-Section record
052	PD-MISC	Procedure Division Miscellaneous
053	-	Reserved for future use.
054	-	Reserved for future use.

Target Code-control Tables

The MetaSuite Generator Target code-control Tables, or output control tables, are code-control tables used for handling target files.

They have the same layout as code-control tables for the source files, but the number of entries and their meaning is different.

Code-control tables for target files are defined in conjunction with the OUTPUT-CONTROL option of the TARGETFILE command, described in the chapter *Definition Language Commands* in the *Metastore Manager User Guide*.

The default target file control tables are:

AU0041	DIF
AU0042	SYLK Files
AU0043	Comma Separated Value (standard delimited type)
AU0044	PRN
AU0045	SAS
AU0051	Variable flat file, line sequential (IBM QSAM)
AU0052	ORACLE
AU0054	SYBASE
AU0055	SQLSERVER
AU0056	BRS/Abacus
AU0057	Sequential fixed target files (IBM QSAM)
AU0058	IMS GSAM
AU0061	Controlled line sequential - variable
AU0067	Controlled line sequential - fixed
AU0068	Controlled line sequential - IMSGSAM
AU0151	Variable flat file, record sequential
AU0157	Fixed flat file, record sequential
AU0161	Controlled record sequential - variable
AU00167	Controlled record sequential - fixed
CSXOUT	Controlled sequential - XML
HTMOUT	HTML file
PAROUT	Parameter file - only one record output
SQLOUT	SQL table as target file (reserved - not supported yet)
MQSOUT	MQ-SERIES output file (reserved - not supported yet)
TR0057	TERADATA sequential fixed
XMLOUT	XML type (simple XML file)
XPCOUT	XML with path control (reserved - not supported yet)

The following table describes the entries in a standard Target code-control Table:

Entry	Function	Generates
001	Contains switches	Header, trailer, data record and title switches
002	Transfer/Report FD	File description for reports, containing header and detail record descriptions
003	Transfer/Report SELECT	SELECT statement
004	Transfer/Report OPEN	Open (output) code
005	Transfer/Report WRT-HDR	Write header record
006	Transfer/Report WRT-FLD-ALPH	Write alphanumeric field
007	Transfer/Report WRT-FLD-NUM	Write numeric field
008	Transfer/Report WRT-FLD-DATE	Write date field
009	Transfer/Report WRT-FLD-GEN	Write fields in general, for most SQL based record types
010	Transfer/Report WRT-TRLR	Write trailer record
011	Transfer/Report CLOSE	Close code
012	Detail FD	File description for detail output, not for reports
013	Detail SELECT	SELECT statement for detail record
014	Detail HDR-STRUCT	Header field definition
015	Detail FLD-STRC-ALPH	Definition of alphanumeric field
016	Detail FLD-STRC-NUM	Definition of numeric field
017	Detail FLD-STRC-DATE	Definition of date field
018	Detail FLD-STRC-GEN	Definition of general field
019	Detail FLD-TLRL	Trailer field definition
020	Detail OPEN	Target file OPEN
021	Detail WRT-HDR	Write header
022	Detail MOV-ALPH	Move instructions for alphanumeric
023	Detail MOV-NUM	Move instructions for numeric
024	Detail MOV-DATE	Move instructions for dates
025	Detail MOV-GEN	Move instructions for general
026	Detail WRT-DATAREC	Write detail record
027	Detail WRT-TRLR	Write trailer record
028	Detail CLOSE	Target file CLOSE
029	WS-MISC	Miscellaneous Working-Storage area
030	LS-MISC	Miscellaneous Linkage-Section area

Entry	Function	Generates
031	PD-MISC	Procedure Division Miscellaneous
032	TDF-WRT-DET	Write detail - initialization

User-written Code-control Tables

The MetaSuite Generator User-written code-control Tables are code-control tables developed by the user.

The format of each entry in a user-written code-control table is the same as the one described above for a standard code-control table. However, a user-written table only includes entries that differ from those in the standard table.

For example, assume that a file requires special processing of an input record immediately after it is read. You might define a code-control table that overrides only the POST-READ entry in the standard table. Your System Administrator input commands might look like this:

```
CHANGE DEFAULT QUOTE
ADD TABLE MYTABL FIXED 12 -
("013 POSTRD")
CHANGE DEFAULT QUOTE
```

The ADD FILE command for the MetaSuite MetaStore might be coded :

```
ADD  FILE  MYFILE  TYPE  SEQUENTIAL  FIXED  120  -
CODE-CONTROL  MYTABL
```

Because the file is a sequential file, the MetaSuite Generator will use the standard code-control table for a sequential file, replacing only entry 13, as specified in table MYTABL.

For more information on how to create your own code-control tables, refer to the section Creating Customized Code-control Tables (page 73)

11.4. COBOL Code Tables

The MetaSuite COBOL code tables are named collections of skeletal COBOL code. Each code table can have from one to many lines of COBOL code. Code tables are maintained on the Generator dictionary using the ADD/ DELETE TABLE commands in this document.

The code table name should be unique in the Generator dictionary.

Within a code table, the COBOL statement must follow COBOL coding conventions. For the most part, you can begin by looking at the standard code table that would be generated for the file.

Be aware that prototype code can be parameterized. A parameter is signalled by a '#' followed by a single number of letter. A general convention used in the system causes file fields to be formatted in a code table as F#1-xxxx. During the actual code-generation process, the '#1' is replaced by the relative number of the file. For example, the name F#1-STATUS in a code table would appear in the program as F01-STATUS when generated for the first file named in the COBOL source program.

The parametrization of code tables is determined by internal generator system code. Should you have any questions about parameterization, contact your IKAN Solutions customer support representative.

For more information on how to create your own COBOL code tables, refer to the section Creating Customized COBOL Code Tables (page 74)

11.5. Code Table Examples

Example 1 - Code Table for Calling I/O Subroutines

In this example, assume you have a file that must be opened, read, and closed using one of your own I/O subroutines. In this case you will need to create a user code-control table for the file that overrides several code-control table entries.

The MIL to add the new code-control tables and code tables would be:

```
CHANGE  DEFAULT  QUOTE
ADD  TABLE  USRTBL  FIXED  12 -
("001  EXCLUDE",-  (do not generate a SELECT)
 "002  EXCLUDE" ,- (do not generate a FD)
 "003  EXCLUDE" ,- (do not generate a FD record layout)
 "005  USRWSM" ,-  (alternate Working-Storage)
 "006" ,-          (generate record layout in W-S)
 "009  USROPN " ,- (call user routine to open file)
 "011  USRRED" ,-  (call user routine to read file)
 "021  USRCLS")    (call user routine to close)

ADD  TABLE  USRWSM  FIXED  56 -
("*      USRWSM  FILE  WORKING-STORAGE  AREA",-
 "A01   F#1 - WS-MISC.",-
 "B02   F#1 - STATUS-CODE.",-
 "C03   F#1 - STATUS-CODE1     PIC X VALUE '0'.", -
 "C03   F#1 - STATUS-CODE2     PIC X VALUE '0'.", -
 "A01   F#1 - USER-PARMS.", -
 "B02   F#1 - USER-CODE        PIC  X.")
```

(The F#1-WS-MISC, F#1-STATUS-CODE, F#1-STATUS-CODE1, and F#1-STATUS-CODE2 fields are status fields required for a sequential file.)

```
ADD  TABLE  USROPN  FIXED  56
("*  USROPN-USER FILE OPEN",-
 "BMOVE '0'  TO  F#1-USER-CODE.",-
 "BCALL  'USROUTIN'  USING  F#1-USER-CODE",-
 "B              F#1-WSREC.")
```

(The name F#1-WSREC is always generated by the MetaSuite Generator for any record layout generated in Working-Storage.)

```
ADD  TABLE  USRRED  FIXED  56 -
("*   USRRED-USER  FILE  READ",-
 "BMOVE 'R'  TO  F#1-USER-CODE.",-
 "BCALL  'USROUTIN'  USING  F#1-USER-CODE",-
 "B               F#1-WSREC.",-
 "BIF  F#1-USER-CODE  EQUAL  'E',-
 "CMOVE  SYS-IOS-EOF-  TO  F#1-IOS1.")

ADD  TABLE  USRCLS  FIXED  56 -
("*    USRCLS-USER  FILE  CLOSE",-
 "BMOVE  'C'  TO  F#1-USER-CODE.",-_
 "BCALL  'USROUTIN'  USING  F#1-USER-CODE",-
 "B               F#1-WSREC.")
CHANGE  DEFAULT  QUOTE
```

Example 2 - Code Table for Calling Reformatting Subroutines

In this example, assume you have a file that can be opened, read, and closed using standard COBOL verbs. However, the record must be decompressed before the program can reference fields in it. In this case you will need to create a user code-control table for the file that overrides the code-control table entries for generating an FD record layout, generating a Working-Storage record layout (for the decompressed record), and post-read processing.

The System Administrator commands to add the new code-control tables and code tables would be:

```
CHANGE  DEFAULT  QUOTE
ADD  TABLE  USRTBL  FIXED  12-
("003  USERFD",- (generate a user FD record)
 "006",-         (generate record layout in W-S)
 "013  POSTRD")  ( call decompress routine)

ADD  TABLE  USERFD  FIXED  56 -
("*  USERFD  FILE  FD  RECORD  AREA",-
 "A01   F#1-FDREC    PIC  X  (300).")

ADD  TABLE  POSTRD  FIXED  56-
("*  POSTRd-POST  READ  COD",-
 "BMOVE  F#1-FDREC  TO  F#1-WSREC.",-
 "BCALL  'USROUTIN'  USING  F#1-FDREC",-
 "B   F#1-WSREC.")
```

(The name F#1-WSREC is always generated by the MetaSuite Generator for any record layout generated in Working-Storage.)

11.6. Creating Customized Code-control Tables

1. Switch to the *MGL Table Maintenance* screen.
   See The MGL Table Maintenance Screen on page 58.

2. Select a standard code-control Table of which the content is similar to the customized Table you require.

3. Replace the Table Name on the first line.
   Choose a name that clearly indicates the purpose of the customized code-control Table.

4. Change the settings as required.
   The format of each entry in a customized code-control Table is identical to the format as described above for a standard code-control Table. However, a user-written table only includes entries that differ from those in the standard table.

5. Click *Save in DDL*.
   The new table will be displayed in the list of available MGL tables.

6. Click *Generate* to make the new table available in the Generator Dictionary.

7. In MetaStore Manager, you can assign the code-control Table to the required Data Source definition.

11.7. Creating Customized COBOL Code Tables

1. Switch to the *MGL Table Maintenance* screen.
   See The MGL Table Maintenance Screen on page 58.

2. Select a standard COBOL Code Table of which the contents is similar to the customized Table you require.

3. Replace the Table Name on the first line.
   Choose a name that clearly indicates the purpose of the customized COBOL code table.

4. Change the settings as required.
   A user-written COBOL code table contains comments, COBOL commands and MTL (MetaSuite Template Language) commands.
   Syntax overview:

Starting character	Generated code begins in column	Description
1	1	COBOL compiler options or JCL code
*	7	COBOL comments
/	7	COBOL page break
A	8	COBOL A-margin code
B	12	COBOL B-margin code
C	16	COBOL column 16 code
D or $	12	Dump code (only generated in debug mode)
-	-	Inactive code
>	-	MetaSuite Template Language command (page 96)
@	-	DDL directives, when creating a new dictionary

5. Click *Save in DDL*.
   The new table will be displayed in the list of available MGL tables.

6. Click *Generate* to make the new table available in the Generator Dictionary.

CHAPTER 12

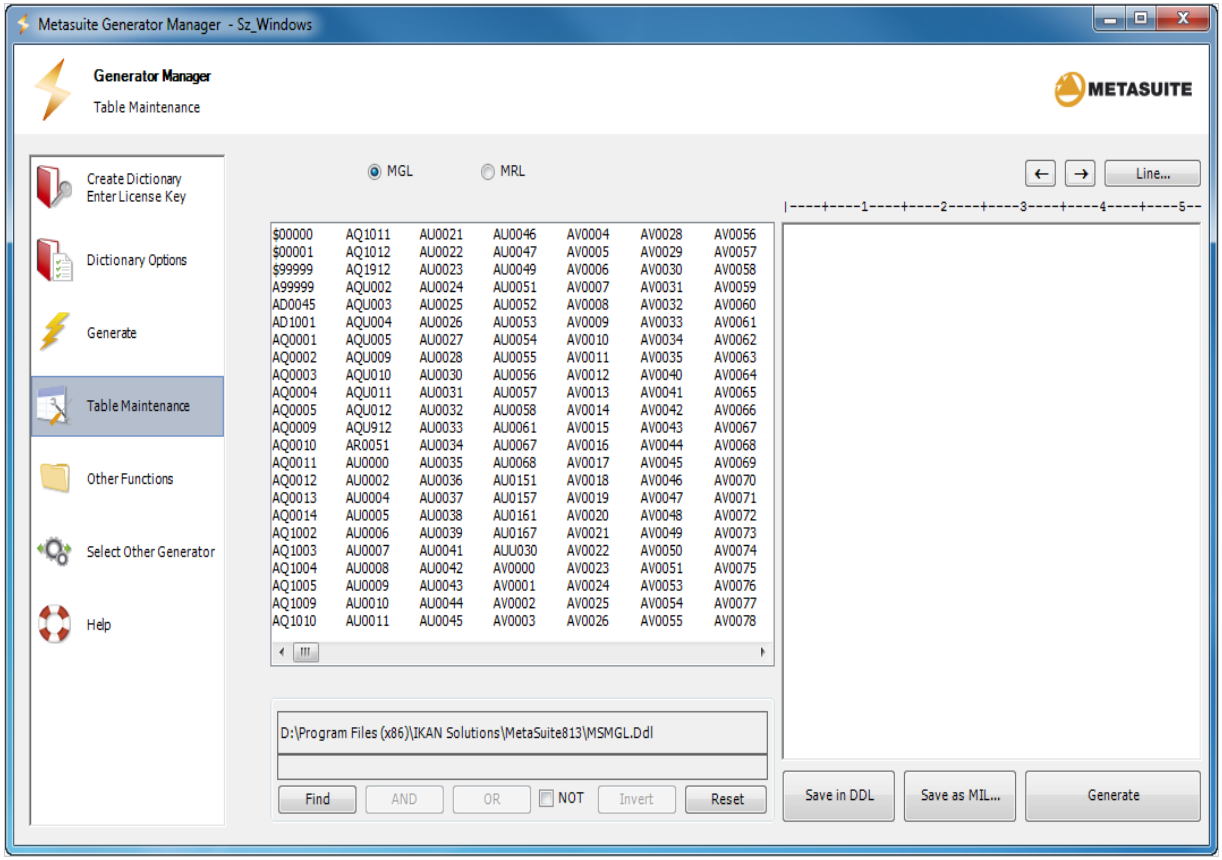
The Table Maintenance Screen - Managing MRL Tables

MRL stands for MetaSuite Run Language. The MSMRL.DDL file (located in the MetaSuite installation folder) is one of three upload files for the Generator Dictionary. The MSMRL.DDL file contains MRL Tables consisting of small pieces of scripts. They describe how the MRL scripts will be generated when a MetaMap model is generated.

Note: It is logical to create a new dictionary after editing the MRL Tables (experienced users only!).

12.1. The MRL Table Maintenance Screen

1. Click the *Table Maintenance* icon in the left column.
   The following screen is displayed:

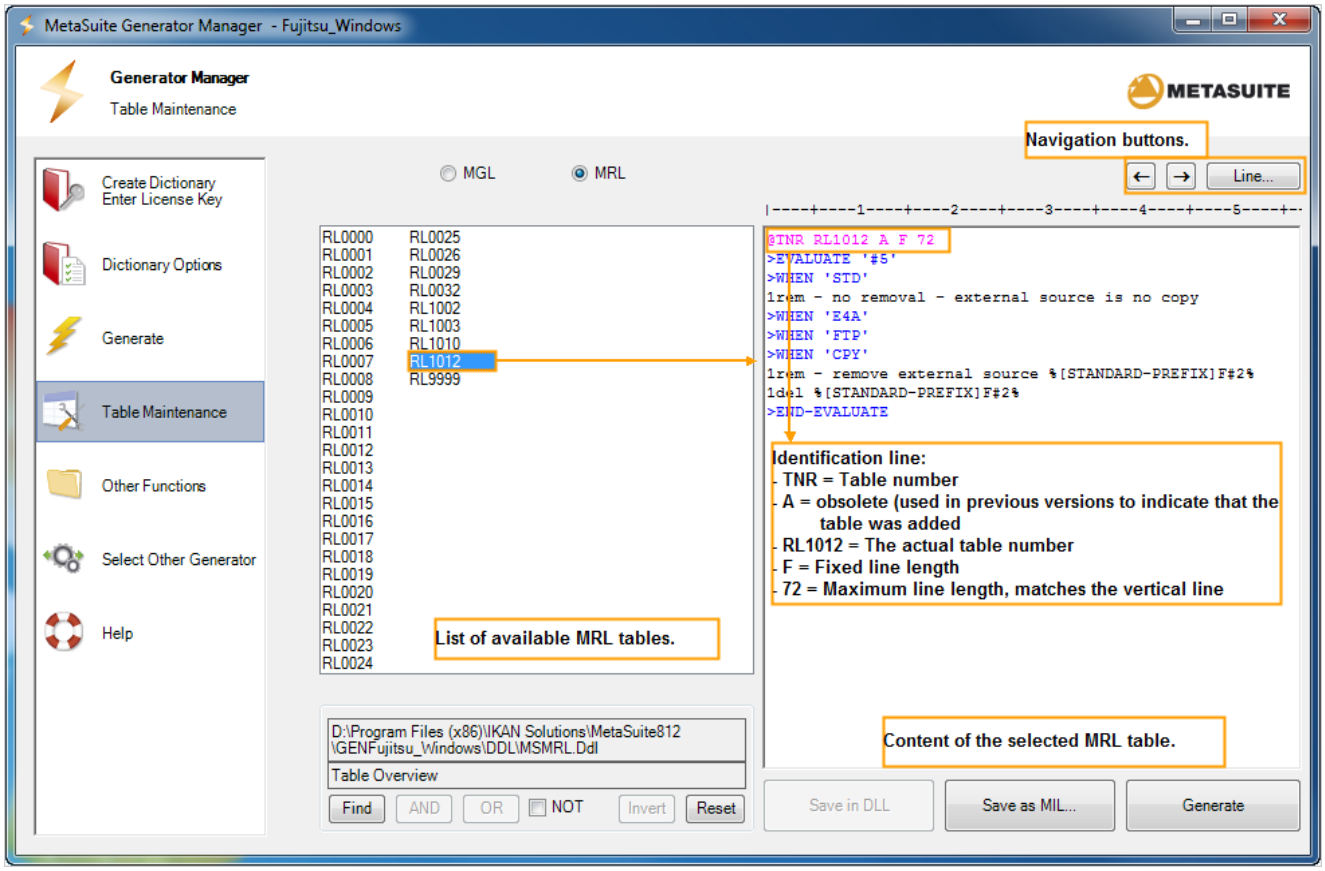


For an explanation of the common parts, refer to the chapter The Table Maintenance Screen - Overview (page 53)

2. Select the *MRL* format option.
   The available MRL tables are displayed.

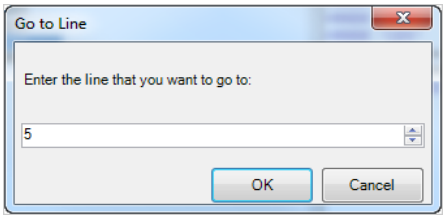
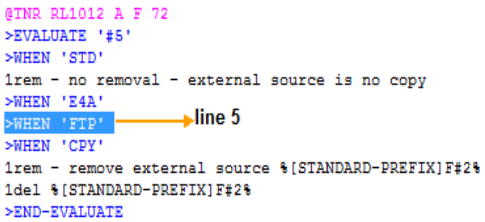
Note: If you want to limit the list of displayed MRL tables, you can define search strings as explained in the chapter The Table Maintenance Screen - Overview (page 53).

3. Click an MRL table to display its contents on the right.
   The following screen is displayed:



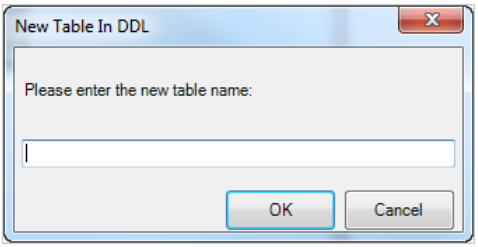
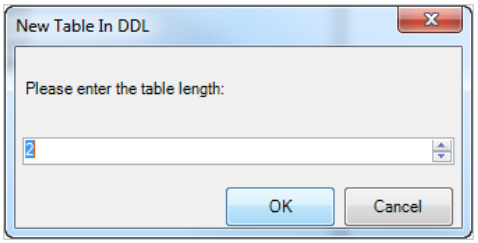
4. Use the Navigation buttons as required.
   The following buttons are available:

Icon	Option	Description
←	Previous table	If you have displayed a sequence of Tables, you can select this option to move backwards in the sequence and to redisplay the previous MRL Table. By repeatedly clicking this option, you can return to the last 10 tables. If there is no more previous table to be shown, the option becomes inactive.

Icon	Option	Description
	Next table	If you have gone backwards in the sequence of displayed tables by means of the *Previous table* option, this option becomes available to go forward in the sequence of displayed MRL Tables. Once you have reached the last (most recent) table in the sequence, the option becomes inactive.
	Go to line number	Click this button to go to a specific line in the currently displayed Table. The following dialog is displayed:  Enter the required line number and click *OK*. As a result, the selected line is highlighted in the file:  **Note**: Lines starting with an @ will not be counted.

5. Edit the code as required.

Note: Right-clicking in the table content's window, displays a pop-up menu with several commands, such as copy, paste, print, ...

Option	Description
New	Right-click the table content's window and select *New* from the pop-up menu. The following dialog is displayed:  Enter the new table name. It must contain exactly 6 characters. Then click *OK*. The following dialog is displayed:  Enter the number of expected lines and click *OK*. The Table is created. You can now enter the required code and save the table to the DDL file by means of the *Save to DDL* option.

6. Once you have performed the required changes, you can use one of the following options: *Save in DDL*, *Save as MIL* or *Generate*.

- Save in DDL

  Click *Save in DDL* to save the currently displayed version of a Table into the MSMRL.DDL file.

  You need to do this:

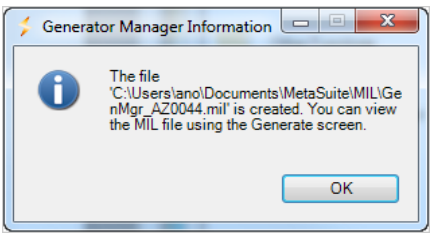
  - after editing an existing file
  - after creating a new file

  The new version of the table is saved in the `MSMRL.DDL` file.

- Save as MIL

  Click *Save as MIL* to save the changes to a table in a new MIL file in the standard MIL file directory.

  The following dialog is displayed:

  
  Generator Manager Information
  The file 'C:\Users\ano\Documents\MetaSuite\MIL\GenMgr_AZ0044.mil' is created. You can view the MIL file using the Generate screen.
  OK

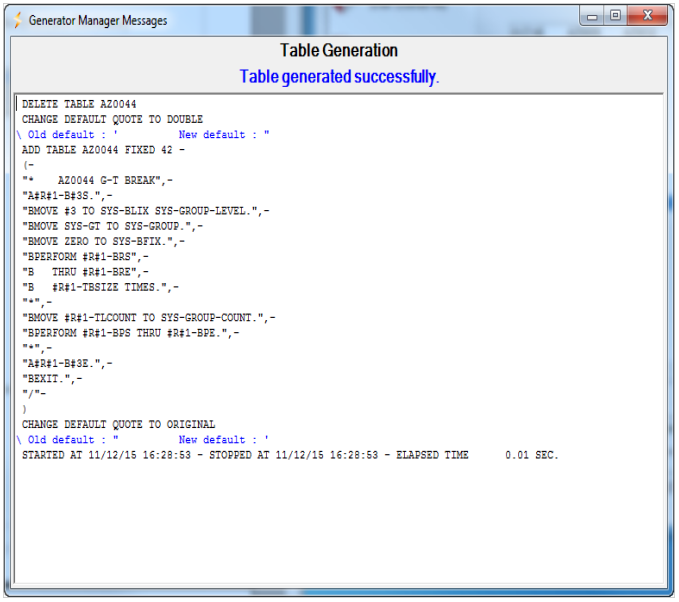
  The file name format is:

  GenMgr_TableName.MIL

  Implement the MIL file to make the changes effective. See Copying Existing MIL Files on page 20.

- Generate

Click *Generate* to make the changes immediately effective in the Generator Dictionary. The following dialog is displayed:



CHAPTER 13
The Other Functions Screen

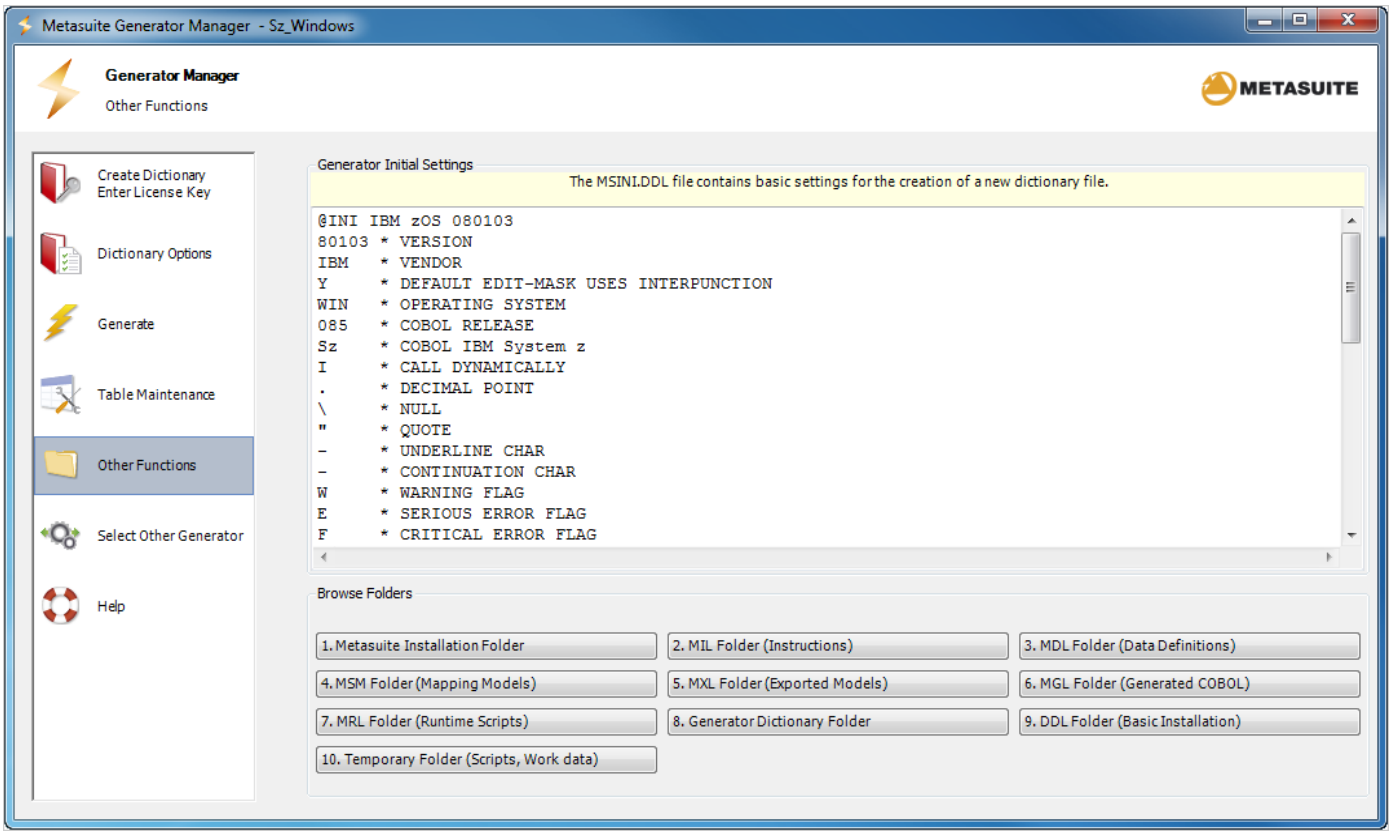
The *Other Functions* screen offers access to the general initial settings and the default browse folders.
Switch to the *Other Functions* screen by clicking the appropriate icon in the left column.
Refer to the following procedures for a detailed description:

- Generator Initial Settings (page 81)
- Browse Folders (page 82)

13.1. Generator Initial Settings

The initial settings are used for the creation of a new dictionary file.

1. Switch to the *Other Functions* screen by clicking the appropriate icon in the left column.
   The Generator Initial Settings are listed in the upper-left corner:



2. Verify the settings.

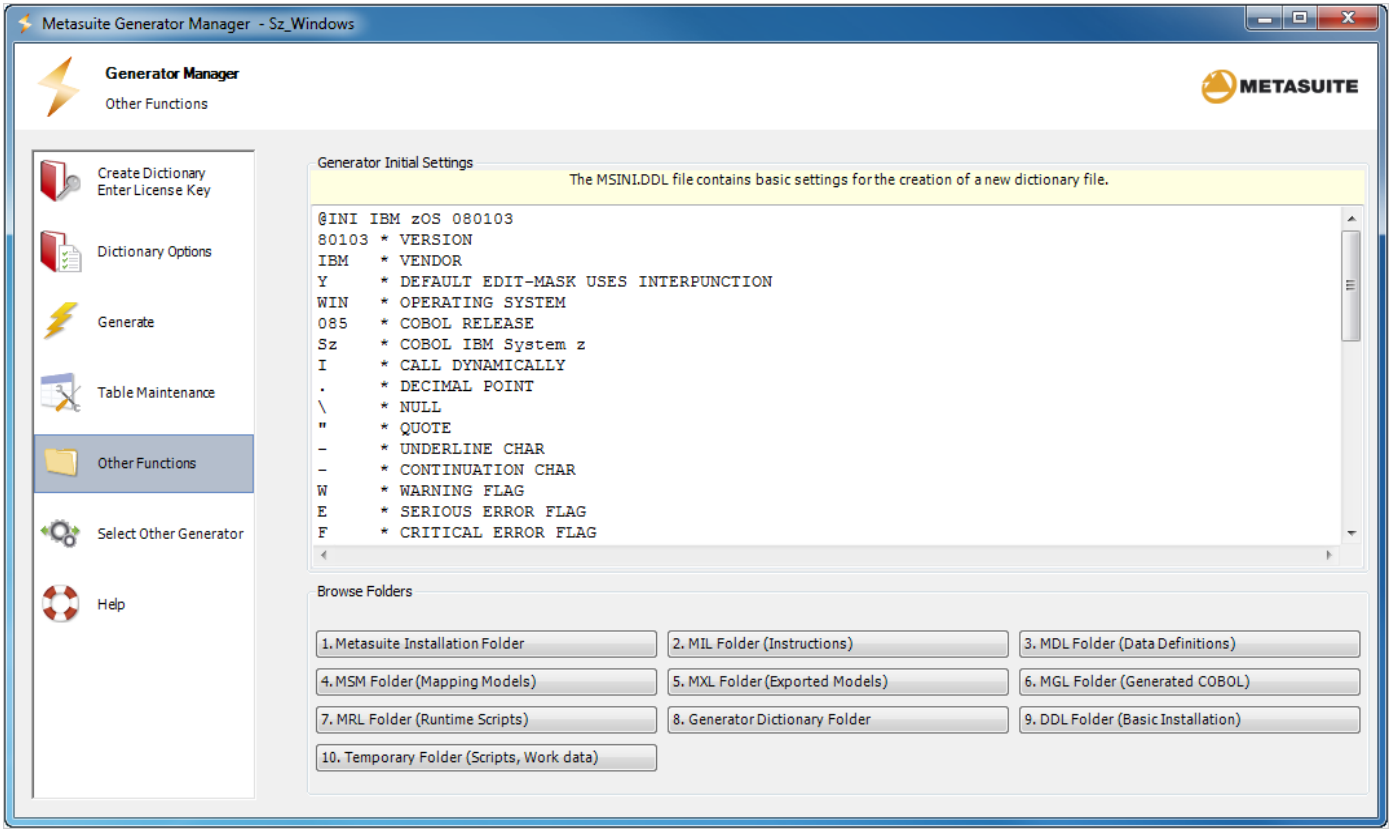
13.2. Browse Folders

This option allows easy viewing of the MetaSuite installation folder and the default folders containing files involved in managing the Generator Dictionary for the selected Generator.

These default folders are specified in the INI Manager.

1. Switch to the *Other Functions* screen by clicking the appropriate icon in the left column.

   The options allowing to view the default folders are listed on lower-left corner.



You can display the contents of the following folders:

Folder	Contents
MetaSuite Installation Folder	Main installation files.
MIL Folder	MIL files contain instructions that can immediately be implemented into the Generator Dictionary. See The Generate Screen - Implementing MIL Instructions on page 18.
MDL Folder	MDL files contain the metadata definitions for Data Sources and Data Targets. See The Generate Screen - Working With MDL Files on page 36.
MSM Folder	MSM files contain the transformation rules defined in a MetaMap model, in a coded form. See The Generate Screen - Working With MSM Files on page 46.
MXL Folder	MXL files contain the transformation rules defined in a MetaMap model, in a readable format. See The Generate Screen - Working With MXL Files on page 48.

Folder	Contents
MGL Folder	MGL files contain the COBOL Source Code that is generated on the basis of a MetaMap model. See The Table Maintenance Screen - Managing MGL Tables on page 57.
MRL Folder	MRL files contain run script commands for the COBOL Source Code (MGL File), which is the result of the generation of a MetaMap Model. See The Table Maintenance Screen - Managing MRL Tables on page 75.
Generator Dictionary Folder	DCT files or Generator Dictionary Files contain the rules used for generating the COBOL source code and the COBOL run scripts on the basis of a MetaMap Model. The Dictionary Files can not be edited directly. For more information, refer to one of the following sections: • Create Dictionary File (page 6) • Implement MIL Instructions (page 18) • Edit MTL Options (page 12) • Edit License Key (page 10) • Edit MRL Tables (page 75) • Edit MGL Tables (page 57)
DDL Folder	There are three DDL files: • MSINI.DDL • MSMRL.DDL • MSMGL.DDL These files are used to create new Generator Dictionaries. DDL files cannot be edited directly. For more information, refer to one of the following sections: • Create New Dictionary File (page 6) • Edit MRL Tables (page 75) • Edit MGL Tables (page 57)
Temporary Folder	The temporary folder is where the generation and compilation takes place.

2. Display the contents of the required folder.

When clicking one of the folder buttons, the folder will be opened with Windows Explorer.
The following table lists the default locations of these folders.

Folder	Default Location
MetaSuite Installation Folder	<InstallationFolder>
MIL Folder (instructions)	<InstallationFolder>\MIL
MDL Folder (data definitions)	<InstallationFolder>\MDL
MSM folder (mapping models)	<InstallationFolder>\MSM
MXL Folder (exported maps)	<InstallationFolder>\MXL
MGL Folder (generated COBOL)	<InstallationFolder>\<GEN>\<MGL
MRL Folder (runtime scripts)	<InstallationFolder>\<GEN>\MRL
Generator Dictionary Folder	<InstallationFolder>\<GEN>\DCT
DDL Folder (basic installation)	<InstallationFolder>\<GEN>\DDL
Temporary Folder (scripts, workdata)	<InstallationFolder>\<GEN>\TMP

Note: If required, you can always modify the default locations in the *MetaSuite.ini* file using the *MetaSuite INI Manager*.

CHAPTER 14
Installation Language Commands

The following Installation Language commandes are available:

- ADD TABLE (page 91)
- CHANGE DEFAULT BUILD (page 86)
- CHANGE DEFAULT CHARACTER-CCSID (page 91)
- CHANGE DEFAULT DATE (page 86)
- CHANGE DEFAULT DECIMAL (page 87)
- CHANGE DEFAULT DYNAMIC (page 91)
- CHANGE DEFAULT EXEC (page 88)
- CHANGE DEFAULT INTERPUNCTION (page 88)
- CHANGE DEFAULT NULL (page 89)
- CHANGE DEFAULT NULLABLE (page 89)
- CHANGE DEFAULT PAGE (page 89)
- CHANGE DEFAULT QUOTE (page 90)
- CHANGE DEFAULT SQL (page 90)
- CHANGE DEFAULT SQL-QUOTE (page 91)
- CHANGE DEFAULT UNICODE-CCSID (page 91)
- COPY TABLE (page 92)
- DELETE TABLE (page 93)
- LIST DEFAULT (page 93)
- LIST TABLE (page 93)
- LIST VERSION (page 94)
- NEW (page 85)
- REMARKS (page 95)

14.1. NEW

```
NEW nnnn
```

The NEW command creates a new Generator dictionary. It must be the first command in a MIL file when you start the generator to perform the initial load of a new Generator dictionary.

Columns 1-3

Required.

Columns 1-3 must contain the keyword NEW.

nnnn

Required.

Columns 5-8 contain a four-digit number (including leading zeros, if appropriate) representing the number of Generator dictionary blocks to be initialized.

14.2. CHANGE DEFAULT BUILD

```
CHANGE DEFAULT BUILD build number
```

The CHANGE DEFAULT BUILD command is used to change the dictionary build number after applying a patch.

Thanks to this command, the generation of a new dictionary can be avoided.

Build Number

Required.

This is a build number between 00 and 99.

Example

Generator Messages
MIL File Generation
MIL file generated without error
CHANGE DEFAULT BUILD 90
\ Old default : 25 New default : 90

14.3. CHANGE DEFAULT DATE

FORMAT 1:

```
CHANGE DEFAULT DATE { 'ISO' | 'EUR' | 'JIS' | 'USA' }
```

The CHANGE DEFAULT DATE command is used to define the date format that is to be used when dates are manipulated in an RDBMS environment.

ISO

All date manipulations in an RDBMS environment are done in the ISO format (YYYY?MM?DD).

When outputting a date, the date separator is a hyphen. (2002-10-30)

EUR

All date manipulations in an RDBMS environment are done in the EUR format (DD?MM?YYYY).
When outputting a date, the date separator is a point. (30.10.2002)

JIS

All date manipulations in an RDBMS environment are done in the JIS format (YYYY?MM?DD).
When outputting a date, the date separator is a hyphen. (2002-10-30)

USA

All date manipulations in an RDBMS environment are done in the USA format (MM?DD?YYYY).
When outputting a date, the date separator is a slash. (10/30/2002)

(century,end-year)

FORMAT 2:

```
CHANGE DEFAULT DATE ( century , end-year )
```

This command is used to define whether years are representing the 20th or the 21st century when the date format has no century included like the data format YYMMDD.

To define the default century used when dates use 2 digit year date formats like YYMMDD, specify two digits that identify the century. For example, 19 specifies the twentieth century and formats years as 19nn; 20 specifies the twenty-first century and formats years as 20nn. The installation default for century is 19.

To define the break point for the default century, specify the end-year as a two digit number that identifies the year the new century should begin to be used when years are specified only as a 2-digit number (e.g., yy). For example, if 19 is defined as the century and the break point for the century is defined as 55, years equal to or higher than the year specified are formatted as 19nn, and numbers less than the end-year specified are formatted as 20nn. The following command sets the century to 19 and the end years to 80 so that dates specified only as yy are assumed to fall in the years between 1980 to 1999 and 2000 to 2079:

CHANGE DEFAULT DATE (19,80)

14.4. CHANGE DEFAULT DECIMAL

```
CHANGE DEFAULT DECIMAL [ TO { POINT | COMMA } ]
```

The CHANGE DEFAULT DECIMAL command is used to switch the default decimal character from a decimal point (.) to a decimal comma (,), and vice-versa.

CHANGE DEFAULT DECIMAL TO COMMA command is used to set the default decimal character to a decimal comma.

CHANGE DEFAULT DECIMAL TO POINT command is used to set the default decimal character to a point

The system is installed with the decimal point character (.) used as the default.

The decimal character has an impact on the generated MGL, telling what the representation of the decimal character needs to be in both source field values and target field values which represent a numeric field but which are declared as an alphanumeric field.

14.5. CHANGE DEFAULT EXEC

```
CHANGE DEFAULT EXEC TO { IMS | RESTARTABLE | Null }
```

The CHANGE DEFAULT EXEC TO IMS command is used to allow restartability under IMS/DC, or under a RDBMS database.

A synonym for "IMS" is "RESTARTABLE". This word is more acceptable when using this option in non-IMS environments.

You can adapt the names of the standard restart routines via the MTL options.

The MTL option CHECKPOINT-CALL contains the name of the COBOL sub program that saves the record information of the last updated record. In non-IMS environments it is set to MSRSTvvv (with vvv=Metasuite version number).

This value can be reset by using the CHANGE DEFAULT EXEC TO NULL command. You do not need to reinitialize the technical dictionary.

This option is supported for OS390 with IMS, and for other Operating systems with RDBMS databases.

14.6. CHANGE DEFAULT INTERPUNCTION

```
CHANGE DEFAULT INTERPUNCTION [ TO { OFF | NO | YES } ]
```

The CHANGE DEFAULT INTERPUNCTION command is used to specify the default edit mask for numeric values (if the CODE property is not set).

If the CODE property is set for a numeric field, the field will be presented with leading zeroes and without interpunction. It will considered as a code, and not as a numeric value that can be counted.

For more information on setting the CODE property, refer to the *Defining Fields* sections for the appropriate Dictionary File in the *MetaStore Manager User Guide*.

If no interpunction value is specified, the interpunction switches from "YES" to "NO", and from any other value to "YES".

OFF

If the INTERPUNCTION option is set to 'OFF', every numeric field will be treated as a code, meaning: no zero suppression and no interpunction.

NO

If the INTERPUNCTION option is set to 'NO', the default edit mask for numeric fields (non-code) will have zero suppression but no interpunction.

(Code numerics will have no zero suppression nor interpunction.)

YES

If the INTERPUNCTION option is set to 'YES', the default edit mask for numeric fields (non-code) will have both zero suppression and interpunction.

(Code numerics will have no zero suppression nor interpunction.)

14.7. CHANGE DEFAULT NULL

```
CHANGE DEFAULT NULL 'Null_Character'
```

The CHANGE DEFAULT NULL command is used to specify the character that will be used to indicate a NULL value within a field in a sequential file.

The Null_Character will be used for both INBOUND and OUTBOUND NULL storage on a field. When this default is not explicitely set, the default value of the Null_Character is set to '\'.

Null_Character

Required. Null_Character should be one character. You should choose your Null_Character carefully in case of INBOUND NULL, since the appearance of the Null_Character on the first position of the field determines that the field has a NULL value.

Following special values are allowed:

SYS-LOW-VALUE or LOW-VALUE : replaces hex "00".

SYS-HIGH-VALUE or HIGH-VALUE : replaces hex "FF".

TAB-CHARACTER : replaces hex "09" .

14.8. CHANGE DEFAULT NULLABLE

CHANGE DEFAULT NULLABLE 'Nullable_Option'

The CHANGE DEFAULT NULLABLE command is used to specify the default NULLABLE option for fields in a sequential file.

If the NULLABLE option is set to 'N', all fields without specific NULL-INDICATOR setting will be treated as NOT-NULLABLE (i.e. NULL-INDICATOR 'NOTNULL').

The Null_Character will not be interpreted for those fields.

If the NULLABLE option is set to 'I', all fields with no specific NULL-INDICATOR setting will be treated as INBOUND-NULLABLE (i.e. NULL-INDICATOR 'INNULL').

Nullable_Option

Required.

Nullable_Option can be 'I' OR 'N'.

'N' is interpreted as NOTNULL, 'I' is interpreted as INNULL.

You should choose your Nullable_Option carefully since the impact of this option is very important.

The standard value for this option is 'N'.

14.9. CHANGE DEFAULT PAGE

```
CHANGE DEFAULT PAGE ( n , m )
```

The CHANGE DEFAULT PAGE command is used to set the default page width and default page length that is used for targets reports when no page settings were specified by the MetaMap Manager model.

`n` represents the default page length (default is set to 80 lines),

`m` represents the default page width (default is set to 80 characters per line).

14.10.CHANGE DEFAULT QUOTE

```
CHANGE DEFAULT QUOTE [TO {SINGLE|DOUBLE|ORIGINAL}]
```

The CHANGE DEFAULT QUOTE command is used to change the default quote character from a single quote (') to a double quote ("), and the reverse.

CHANGE DEFAULT QUOTE TO SINGLE changes the default quote character into a single quote.

CHANGE DEFAULT QUOTE TO DOUBLE changes the default quote character into a double quote.

CHANGE DEFAULT QUOTE TO ORIGINAL resets the default quote character to its original value at the time that the generating process was started.

Most systems are installed with the single quote character used as the default.

Most code tables and patches however use the double quote as default.

The default setting can be found in the MSINI.DDL file that is used to initialize the MetaSuite Generator dictionary, and can be listed by the command

"LIST DEFAULT ALL"

All non-numeric constants within your MetaSuite generator models must be delimited with the default quote character.

MetaMap Manager will use the single quote as delimiter for non-numeric constants.

The main use of the CHANGE DEFAULT QUOTE command is to temporarily change the default quote character, prior to adding or changing one or more table definitions.

It is recommended that you do not alter this on a permanent basis with this command, because if you regenerate the dictionary, the original default setting will be reset to the content that can be found in the MSINI.DDL file.

Changing the default quote character before entering the ADD TABLE command allows you to embed single quote characters in the text of any table entry being added or replaced. After all of the table commands have been entered, you should then change the default back by coding CHANGE DEFAULT QUOTE TO ORIGINAL command.

The following sequence of commands illustrates the use of the CHANGE DEFAULT QUOTE command when modifying library tables:

CHANGE DEFAULT QUOTE TO DOUBLE

DELETE TABLE SALES-HIST-READ

ADD TABLE SALES-HIST-READ 4 -

(-

"BCALL ""HSTRDR03"" USING F#1-RD0001.",-

"BIF F#1-RD0001 = SPACE EXIT."-

)

CHANGE DEFAULT QUOTE TO ORIGINAL

14.11.CHANGE DEFAULT SQL

```
CHANGE DEFAULT SQL DIALECT [TO]
{
DB2_400 | DB2_MVS | DB2_LUW | DB2_2 | DB2_VSE | INFORMIX | INGRES | MYSQL
| ODBC | ORACLE | RDB | SESAM | SQLSERVER | SYBASE | TERADATA
}
```

The CHANGE DEFAULT SQL DIALECT command is used to set the preferred SQL dialect to be used when embedded SQL is generated and no specific SQL DIALECT was chosen by the MetaMap Manager.

The following sequence of commands illustrates the use of the CHANGE DEFAULT SQL DIALECT command when modifying library tables:

CHANGE DEFAULT SQL DIALECT TO ORACLE

14.12.CHANGE DEFAULT SQL-QUOTE

The CHANGE DEFAULT SQL-QUOTE command is used to set the preferred SQL quotes to be used when embedded SQL is generated and no specific SQL-QUOTE was chosen by the MetaMap Manager.

The following sequence of commands illustrates the use of the CHANGE DEFAULT SQL-QUOTE command when modifying library tables:

CHANGE DEFAULT SQL-QUOTE [TO {SINGLE|DOUBLE|ORIGINAL}]

CHANGE DEFAULT SQL-QUOTE TO SINGLE changes the default quote character for SQL commands into a single quote.

CHANGE DEFAULT SQL-QUOTE TO DOUBLE changes the default quote character for SQL commands into a double quote.

CHANGE DEFAULT SQL-QUOTE TO ORIGINAL resets the default quote character for SQL commands to its original value at the time that the generating process was started.

14.13.CHANGE DEFAULT DYNAMIC

The MetaSuite program can be generated in three ways:

1. the runtime modules are called in a static way. (Static binding)
2. the runtime modules are called dynamically. (Dynamic linking)
3. the runtime modules are called internally: the runtime modules form part of the source file. Everything is compiled and linked in one go.

14.14.CHANGE DEFAULT CHARACTER-CCSID

Default CCSID for each single-byte character field.

14.15.CHANGE DEFAULT UNICODE-CCSID

Default CCSID for each Unicode field.

14.16.ADD TABLE

```
ADD TABLE table-name FIXED entry-size ('table-entry',...)
```

There are three basic types of library tables: system tables, code-control tables, and prototype code tables.

System tables are used for such things as reserved words, error messages, and tables of system-dependent parameters. code-control tables for source files are defined in conjunction with the CODE-CONTROL option of the ADD FILE commands. Prototype code tables contain prototype COBOL code (referenced in code-control tables) that will be inserted into generated COBOL programs, usually with some modification. The ADD TABLE command is used to add a table to the Generator dictionary. It can be used to add any type of table, but is used typically only to define code-control tables and their associated prototype code tables.

Table-name

Required. Table-name is the name of the table being defined to the library. The name must be unique within the library; may be up to 6 characters in length; must begin with an alphabetic character; and may contain the characters A-Z, 0-9, and embedded hyphens.

Entry-size

Required. The entry-size option indicates the maximum length, in characters, of any one entry in the table. Any entries that are shorter than entry-size will be padded on the right with blanks. The maximum entry-size should be no longer than 72.

Table-entry

Required. Table-entry is a character string, in any of three possible formats. At least one table-entry must be defined (to provide code that will be expanded into the generated program). Each table-entry must be enclosed in quotes and separated from the next entry by a comma. If any table-entry contains embedded quote characters, use the CHANGE DEFAULT QUOTE command prior to executing the ADD TABLE, to change the default quote character to a double quote. Then, enclose each table-entry in double quote characters, and revert to the default single quote character following the ADD TABLE command by coding another CHANGE DEFAULT QUOTE command. Each of the three possible formats for table-entry is described in more detail below.

System tables are never added to the library unless a generator software correction directs you to delete an existing system table and replace it with a new version of the same table. Extreme care must be used when coding system tables entries. Entries that are coded incorrectly in a system table may render your library unusable.

14.17.COPY TABLE

```
COPY TABLE table-name
```

The COPY TABLE command is used to copy one or more Generator dictionary tables to an output command file. The file to contain the copied commands is MSCOPY.MIL, and can be found in your TEMP variable.

table-name

Required. Table-name is the name of the Generator dictionary table to be copied to the output command file. You can request system and/or user-coded tables. For example, to copy the definition of the table HSTCTL to the output command file, you would use the following command:

COPY TABLE HSTCTL

14.18.DELETE TABLE

```
DELETE TABLE table-name
```

The DELETE TABLE command is used to delete a table from the Generator dictionary. Be careful not to delete any of the system tables from the library, as this may render the system unusable. You can ensure your ability to recover from such an error by using the COPY TABLE command (to copy the definition of the table to be deleted to a backup file) prior to executing the DELETE TABLE command. Should it become necessary, then, you could restore the table definition from the backup file.

Table-name

Required. Table-name is the name of the Generator dictionary table to be deleted. For example, to delete the table named HSTCLS from the library, you might use the following command:

DELETE TABLE HSTCLS

14.19.LIST DEFAULT

```
LIST DEFAULT
```

The LIST DEFAULT command is used to display a list of all the defaults that apply to the Generator dictionary.

Example

```
LIST DEFAULT \Character Set :
\    Decimal Point       .
\    Quote               '
\    Null                \
\Nullable :                 NOTNULL
\Maximum line length :      00080
\Maximum page length :      00080
\SQL Dialect :              ODBC
\COBOL Calls :              Dynamic
\Edit Mask Interpunction :  Yes
\Executive :                Null
\Default Date Format :      ISO       (19,00)
```

14.20.LIST TABLE

```
LIST TABLE table-name
```

The LIST TABLE command is used to list a table in the Generator dictionary. This command is particularly useful to produce a hard copy listing of each Generator dictionary table before modifying or deleting it, in the event that it becomes necessary to restore the table to its original state.

Table-name

Required. Table-name identifies the name of the Generator dictionary table which you want to be listed.

Example

```
LIST TABLE AV0045
\ TABLE NAME REPORT
\ TABLE LENGTH BLOCK NO OFFSET
\ ***** ****** ******** ******
\ AV0045 0048 0005 0603
\ ENTRIES:
\ 001 >EVALUATE '[SYS-COBOL]/[SYS-OPERATING-SYSTEM]'
\ 002 >WHEN 'VS/VSE'
\ 003 >WHEN 'VS/MVS'
\ 004 >WHEN 'SZ/ZOS'
\ 005 >WHEN 'SZ/WIN'
\ 006 >WHEN 'SZ/UX'
\ 007 B ASSIGN TO DA-I-[STANDARD-PREFIX-NO-DD]F#1
\ 008 >WHEN 'VA/UX '
\ 009 >WHEN 'VA/WIN'
\ 010 B ASSIGN TO [STANDARD-PREFIX-NO-DD]F#1
\ 011 >WHEN 'O4/OS4'
\ 012 B ASSIGN TO DISK-[STANDARD-PREFIX-NO-DD]F#1
\ 013 >WHEN 'C2/BS2'
\ 014 B ASSIGN TO "[STANDARD-PREFIX-NO-DD]F#1"
\ 015 >WHEN 'DG/VMS'
\ 016 >WHEN 'AC/VMS'
\ 017 B ASSIGN TO [STANDARD-PREFIX-NO-DD]$F#1
\ 018 >WHEN 'MF/UX '
\ 019 >WHEN 'MF/WIN'
\ 020 B ASSIGN TO DYNAMIC F#1-FILENAME
\ 021 >WHEN OTHER
\ 022 B ASSIGN TO F#1-FILENAME
\ 023 >END-EVALUATE
\
STARTED AT 13/07/30 16:48:28 - STOPPED AT 13/07/30 16:48:28 - ELAPSED TIME 0.02 SEC.
```

14.21.LIST VERSION

`LIST VERSION`

The LIST VERSION command is used to list the version and build number of the current generator, and the version plus build number of the Generator dictionary.

The output of this command contains also the build number requirements of the generator and the Generator dictionary: a certain build of a generator corresponds with a certain build number of the dictionary, and vice versa.

Example

```
LIST VERSION
  GENERATOR VERSION IS 07020
  GENERATOR BUILD NUMBER IS 5
  MINIMUM DICT BUILD FOR THIS GENERATOR IS 4
  DICTIONARY VERSION IS 07020
  DICTIONARY BUILD NUMBER IS 4
  MINIMUM GENERATOR BUILD FOR THIS DICT IS 55
```

14.22.REMARKS

`REMARKS text`

The REMARKS command allows you to insert lines of comments. Liberal use of remarks is recommended, to document the command file. Text consists of any amount of descriptive text. For example, the following REMARKS command spells out very clearly what's happening in the file procedure that follows:

REMARKS Choose a Source RDBMS

When coding multiple lines of comments, be sure to include the continuation character at the end of each line of text to be continued:

REMARKS Choose a -
Source RDBMS

CHAPTER 15

MetaSuite Template Language

The Generator library is a collection of COBOL code tables required to translate MetaMap Models (MSM) into a COBOL application for the requested operating system.

Next to COBOL commands, these tables can also contain lines with logic written in the *MetaSuite Template Language (MTL)*. These lines contain the > character in column 1 followed by the actual MTL command. The Generator interprets this MTL logic before writing lines onto the generated MGL and MRL scripts.

While using MTL, you can define variables, make conditional structures and build loops. Code table variables can be included in MTL instructions and MTL variables can be included in the COBOL code of the code tables.

Many tables contain variables or character strings that are replaced by the generator during the generation process. There are two types of variables:

- MTL variables (enclosed by square brackets).
- Generator variables (composed of the # sign followed by a single character). Their value is not defined by MTL code, but provided by the generator. Their value changes from table to table. The replacement value can be zero to 32 characters.

Refer to the following procedures for a detailed description:

- MTL Variables, Expressions and Functions (page 96)
- Generator Variables (page 101)
- MTL Commands (page 102)
- MTL System Variables (page 118)
- SourceFile-specific MTL Variables (page 120)
- SourceRecord-related MTL Variables (page 121)
- TargetFile-related MTL Variables (page 122)
- TargetRecord-related Variables (page 123)
- Field-related MTL Variables (page 124)

15.1. MTL Variables, Expressions and Functions

Variables are pieces of memory space that can be accessed using the variable name. The organization of the variable content is defined by the type of the variable.

MTL expressions are combinations of variables, numbers and operators. They are used to define or calculate the content of an MTL variable and can also be used in comparisons.

MTL Variables

The number of MTL variables used for generating a single program is limited to 1000.

The combination of the syntax of a generator variable and the syntax of an MTL variable is not allowed. For example: `#[A] or [[B]]` is not allowed.

The following table lists the characteristics of an MTL variable.

Variable	Description
Name	The name must respect the following rules: • maximum length: 32 characters • first character must be alphabetic • forbidden characters: SPACE = # [and] • characters to be avoided: + - / and * • the name is case-insensitive.
Type	An MTL variable can be: • numeric: integers from –99999999 to +99999999 • alphanumeric: length can be from 0 to 58 bytes.
Content	The content of an MTL variable can be retrieved in any code table entry by putting the variable name between square brackets ("[" and "]"). Numeric variables will be trimmed by the generation process: trailing blanks will be removed. Alphanumeric variables will not be trimmed: trailing blanks will not be removed. The content of the generator variables and MTL variables will be replaced by their values before the MTL syntax is checked!
Scope	The scope of a variable is not limited to one table. The interpretation sequence is equal to the lines appearing in the resulting MGL and MRL output. MTL variables defined by one table are kept in memory, so that all tables at a later stage can invoke them. (scope = global) This means that a variable created on line 100 of the MGL will still be available when using it on line 2000 of the MGL. Furthermore, it will also exist when the MRL is generated.

Temporary MTL Variables

MTL variables with a name that starts with "@" are temporary variables.

The scope of a temporary variable is limited to the table it is created for, or to one CALL-level lower. In order to use the temporary MTL variable from a higher level, the leading "@" character has to be replaced by "?".

Sample:

```
@TNR MTLTST A F 70
>*******************************************************
>* TABLE : MTLTST *
>* PURPOSE : TESTING MTL *
>* DRIVER : PPTGGS *
>* CHANGED : FIB 30/11/2011 *
>*******************************************************
>* THIS TABLE HAS NO OTHER PURPOSE THAN FOR TESTING MTL
>* INSTRUCTIONS.
>* IT IS THE FIRST TABLE OF THE GENERATION.
>*******************************************************
>CALL TABLE1

@TNR TABLE1 A F 70
>SET @NUMBER = 1000
```

```
>SET @TEXT = 'THE QUICK BROWN FOX JUMPS...'
* @NUMBER = [@NUMBER]
* @TEXT = [@TEXT]
>CALL TABLE2
* @NUMBER = [@NUMBER]
* @TEXT = [@TEXT]

@TNR TABLE2 A F 70
*** ENTERING TABLE2 ***
>SET @NUMBER = [?NUMBER]+1
>SET @TEXT = '...OVER THE LAZY DOG'
* @NUMBER = [@NUMBER]
* ?NUMBER = [?NUMBER]
* @TEXT = [@TEXT]
* ?TEXT = [?TEXT]
*** LEAVING TABLE2 ***

RESULT:
      * @NUMBER = 1000
      * @TEXT = THE QUICK BROWN FOX JUMPS...
      *** ENTERING TABLE2 ***
      * @NUMBER = 1001
      * ?NUMBER = 1000
      * @TEXT = ...OVER THE LAZY DOG
      * ?TEXT = THE QUICK BROWN FOX JUMPS...
      *** LEAVING TABLE2 ***
      * @NUMBER = 1000
      * @TEXT = THE QUICK BROWN FOX JUMPS...
```

MTL Expressions

The following table lists the MTL expression types.

Expression type	Description
Numeric	Numeric expressions are combinations of numeric characters and operators (+, -, * and /). The Generator will perform a calculation on this expression, and the result of this process will be taken in account. Numeric variables can be used when surrounding them by square brackets. Example: **4+5*6-7*[NUM]/8**
Alphanumeric	Alphanumeric expressions begin with a single quote or double quotes, and end with the same type of quote. If the alphanumeric expression (string) contains one of the enclosing quotes, this quote has to be doubled. Example: **"ALPHA IS [ALPHA]"**
Variable names	If the expression is not a numeric expression nor a string, and not the word "FUNCTION", the Generator assumes that the expression is a variable name. Example: **AZERTY(1)**
Function	If the expression starts with the word "FUNCTION" OR "F:", an MTL function will be executed, and the output of that function will be taken as value. Example: **FUNCTION MID("BRUSSELS BY NIGHT",10,2)**

MTL Functions

The syntax of the FUNCTION command is very strict:

- Just after the word "FUNCTION", there can be only one space, followed by the function name.
- The word "FUNCTION" can be replaced by "F:"
- After the function name, round brackets surround the parameters.
- The parameters inside the brackets are separated by one delimiting character, which can be a comma or a space.

The following table lists the MTL functions.

Function	Description
LENGTH	**The function LENGTH** returns the length of a string. Syntax: **FUNCTION LENGTH("string")** The use of the surrounding quotes is mandatory! Example: **>SET X="TEST"** **>SET L=FUNCTION LENGTH("[X]")** Result: L = 4
MIDDLE	The function **MIDDLE** returns a part of a string. Syntax: **FUNCTION MIDDLE("string",startposition[,length])** Where: • **string** = the main string from which we want to get a substring • **startposition** = the starting position of the part. This parameter must be a number between one and the length of the main string. • **length** = the length of the substring. This length must be zero or a positive number. If the length parameter is omitted, the length will be the remaining length of the main string, starting from the start position The use of the surrounding quotes in the first parameter is mandatory! Example: **>SET X="TEST123"** **>SET Y=FUNCTION MIDDLE("[X]",3,2)** **>SET Z=FUNCTION MIDDLE("[X]",5)** Result: • Y = ST • Z = 123
LEFT	The function **LEFT** returns the left part of a string. Syntax: **FUNCTION LEFT("string",length)** Where: • **string** = the main string from which we want to get a substring • **length** = the length of the substring. This length must be zero or a positive number. The use of the surrounding quotes in the first parameter is mandatory! Example: **>SET Y=FUNCTION LEFT("TEST123",3)** Result: Y = TES

Function	Description
RIGHT	The function **RIGHT** returns the right part of a string. Syntax: **FUNCTION RIGHT("string",length)** Where: • **string** = the main string from which we want to get a substring • **length** = the length of the substring. This length must be zero or a positive number. The use of the surrounding quotes in the first parameter is mandatory! Example: **>SET Y=FUNCTION RIGHT("TEST123",5)** Result: Y = ST123
INSTRING	The function **INSTRING** searches for a substring in a string. Syntax: **FUNCTION INSTRING("string1","string2"[,startposition])** Where: • **string1** = the main string from which we want to get a substring • **string2** = the substring, which may be a part of the main string • **startposition** = the starting position for the search operation. This position must be a positive number. If omitted, the starting position is one. The use of the surrounding quotes in the first and second parameter is mandatory! Example: **>SET Y=FUNCTION INSTRING("BRUSSELS","SEL",3)** **>SET Z=FUNCTION INSTRING("BRUSSELS","XYZ")** Results: • Y = 5 (search string starts at position 5). • Z = 0 (search string not found)
UPPER	The function **UPPER** converts all alphabetic characters in a string to uppercase. Syntax: **FUNCTION UPPER("string")** The use of the surrounding quotes is mandatory! Example: **>SET Y=FUNCTION UPPER("Brussels")** Result: Y = BRUSSELS
LOWER	The function **LOWER** converts all alphabetic characters in a string to lowercase. Syntax: **FUNCTION LOWER("string")** The use of the surrounding quotes is mandatory! Example: **>SET Y=FUNCTION LOWER("Brussels")** The use of surrounding quotes is mandatory! Result: Y = brussels
IS-SET	The function **IS-SET** is used to test whether a variable has been declared or not by the SET instruction. Syntax: **FUNCTION IS-SET(variable-name)** The use of the surrounding brackets is not mandatory. Example 1: **>SET Y=FUNCTION IS-SET(SYS-QUOTE)** Result: Y = "TRUE" Example 2: **>SET Z=FUNCTION IS-SET(UNKNOWN-VARIABLE)** Z = "FALSE".

Function	Description
VAR-TYPE	The function **VAR-TYPE** is used to test whether an MTL variable is considered as being numeric, alpha-numeric or not defined. The result can either be "ALPHA", "NUM", or "NULL". Syntax: **FUNCTION VAR-TYPE(variable-name)** The use of the surrounding brackets is not mandatory. Example 1: **>SET Y=FUNCTION VAR-TYPE(SYS-QUOTE)** Result: Y = "ALPHA" Example 2: **>SET Z=FUNCTION VAR-TYPE(UNKNOWN-VARIABLE)** Z = "NULL".
VAR-LENGTH	The function **VAR-LENGTH** is used determine the size of an MTL variable. Syntax: **FUNCTION VAR-LENGTH(variable-name)**

QUOTE	The function **QUOTE** is used to double the quotes within an MTL variable. This feature can be used in order to embed a value in a string.
	Syntax:
	FUNCTION QUOTE(variable-name)
	Sample:
	`>  SET @DECL-LEN  = F:LEN(@DECL)`
	`>  SET @DECL-QUOT = F:QUOTE (@DECL)`
	`B10  FILLER PIC X([@DECL-LEN]) VALUE "[@DECL-QUOT]"`

15.2. Generator Variables

Generator variables are not defined by MTL code. They are provided by the generator. Their value changes from table to table. The replacement value can be zero to 32 characters.

Generator variables are supplied by the generator. Their names always start with the # character, followed by one of the following characters "123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ.'[23]0".

Generator variables can not only be used in the COBOL code, but also in MTL expressions. The length of those variables is limited to 54 characters.

The generator variable "##" is used as a field level indicator.

Value	Description
#1 to #9	Generator variable number 1 to number 9
#A to #U	Generator variable number 10 to number 30
##	The # character
#£	The sequence number. This is the number indicating how many times this variable has been called.
#µ	The current COBOL line number within the MGL (COBOL) or MRL (scripting language).
#$	Field level indicator: "___02_" for restartable/IMS code, and "_01_" for other code
©	The version of MetaSuite (3-digit code)

The content of the variables is always trimmed by the generator: trailing blanks are removed.

Note: The content of the generator variables and of the MTL variables will be replaced by their values before the MTL syntax is checked!

15.3. MTL Commands

The following table lists the available MTL Commands. Refer to the dedicated sections for a more detailed description.

MTL Command	Short description
CALL table (page 104)	The *CALL table* command makes it possible to insert table logic from one table into another table. This is interesting if the same piece of logic returns on different occasions. The CALL instruction is also used for simplifying complicated tables.
DUMP (page 105)	The *DUMP* statement is an instrument used to debug the generated code.
EVALUATE structures (page 105)	*EVALUATE* structures are used when the action to be taken depends on different variable values. The EVALUATE statement evaluates an expression and executes the statements matching the evaluation result.
EXIT table (page 107)	The *EXIT* instruction allows exiting a table: • in order to return to the calling table, if nesting level is not zero. For the EXIT statement the same rules are applied as for the GOTO statement. • In order to proceed with the next table (if nesting level is zero).
Fault Message handling (page 107)	In the MetaSuite Template Language, you can program your own fault messages and stop the generating process, if required.
FOR-NEXT loops (page 110)	*FOR-NEXT* loops are used to perform structured loops. They help the table programmer to program loops more easily.
FREEZE and UNFREEZE (page 111)	The *FREEZE* command stops the code table interpreter (of which the MTL interpreter is a part) from performing some standard actions. *UNFREEZE* restarts the normal code line treatment.
GOTO (page 111)	*GOTO* is a jump command that allows skipping or repeating certain MTL commands and COBOL code in the generated MGL.
IF structures (page 113)	*IF* structures are necessary to perform condition dependent actions.
IMPLEMENT (page 114)	The *IMPLEMENT* command is used in order to call a section at a later stage.
NESTED IF (page 114)	It is possible to nest multiple **IF** and **EVALUATE** blocks.
REMARK (page 114)	The *REMARK* command has an informational purpose. It provides the user with the possibility of putting some comment in the MTL code.
SET (page 115)	The *SET* command creates or modifies an MTL variable.
SKIP (page 115)	*SKIP* is a jump command allowing to skip or repeat certain MTL commands and COBOL code in the generated MGL.

MTL Command	Short description
TRACE (page 116)	The *TRACE* command is used for tracing or debugging the MTL code. It provides more information about the logic followed by the MTL interpreter.
UNSET (page 117)	The *UNSET* command removes an MTL variable.

Notation conventions

This chapter describes the meta syntax used to describe the different MTL commands. Each command is composed of keywords, optional keywords and coding rules.

The sample syntax below demonstrates the notation conventions used for describing an MTL command.

COMMAND field-name
[OPTION-ONE (user-value,...)]
[OPTION-TWO {A | B | C}]
[OPTION-THREE]°
[OPTION-FOUR]¹
[RULE text]

Description:

- Keywords:

  Keywords are considered reserved words and may not be used as WorkField or MetaStore names. In our example, RULE is a keyword.

- Variable user-supplied information:

  Variable user-supplied information is written in lowercase. In our example, "field-name", "user-value" and "text" all indicate information to be supplied by the user.

- Square brackets ([ ]):

  Square brackets indicate optional components of the command, i.e. options that you can include or not, as appropriate to your processing. Do not include the brackets in your program code. With few exceptions, options may be coded in any order. **RULE**, which is an option on every command, must be last, if used.

  If the closing square bracket is followed by a zero in superscript, this option may be repeated several times.

  If the closing square bracket is followed by the cipher *1* in superscript, this option may be repeated several times, and must be used at least one time.

- Parentheses:

  Parentheses should be included in program code exactly where shown in the syntax. Parentheses are used in program code to enclose lists of items, as illustrated in the previous example.

  A comma and ellipsis (three dots) following an item indicates that a list of one or more similar items is expected, as shown in our example above for OPTION-ONE.

  If multiple items are coded, each item should be separated from the next by a comma.

- Curved brackets ({ }):

  Curved brackets enclose alternative choices, one of which must be selected. Within the curved brackets, vertical bars separate the alternative choices.

  In our example, if you choose to code OPTION-TWO, you must include either the keyword "A", "B" or "C". Do not include the brackets in your program code.

CALL table

The *CALL table* command makes it possible to insert table logic from one table into another table. This is interesting if the same piece of logic returns on different occasions. The CALL instruction is also used for simplifying complicated tables.

Syntax:

> **CALL table-name [label]**

Where:

- **table-name**:

  This is the name of the table that will be called. This table will be named "the called table", while the table containing the CALL instruction will be referred to as "the calling table".

- **label**

  If the user wants to skip certain lines in the called table, he can specify a label within the called table. The lines before that label will be skipped.

The MTL variables and the table specific variables (#1, #A, etc…) are available in the called table. All variables have a so-called "global" character. "Local" variables are not supported by MTL.

You can do a CALL within an already called table. CALL nesting can go up until 16 levels. The nesting level is the number of nested calls that has been performed. The nesting level is zero for the tables that are directly called by the generator.

The number of spaces between the > character and the **CALL** statement is important because the alignment mechanism adds this number of spaces to the starting column of each line generated by the called table! Only comment lines do not follow that rule.

DUMP

The *DUMP* statement is an instrument used to debug the generated code.

If the Generator encounters a *DUMP* statement, the following actions will be performed:

- The content of all MTL variables will be written in the MGL code
- The name of the last called sub-program in the Generator will be displayed in the MGL code
- The content of the generator variables will be listed in the MGL.
- The generating process will be stopped, but the MGL code will not be deleted.
- The MGL output contains a message in the following format:

  *\TBG568E 1 ** DUMP STATEMENT EXECUTED ** GENERATION INTERRUPTED **

Syntax:

>DUMP

Results:

The generated COBOL Source code will contain a section in the following format:

```
********************************** DUMP OUTPUT ************************
SYS-GENERATOR-VERSION              = '70201'
SYS-OPERATING-SYSTEM               = 'MVS'
SYS-COBOL-VERSION                  = '085'
SYS-COBOL                          = 'VS'
SYS-COMMA                          = ' '
SYS-DECIMAL-POINT                  = '.'
SYS-NULL                           = '\'
SYS-QUOTE                          = ''''
SYS-DATE-FORMAT                    = 'ISO'
SYS-NULLABLE                       = 'N'
SYS-SQL-DIALECT                    =          0
SYS-EXEC                           = 'NULL'
SYS-DATE-WRITTEN                   =  20020122
SYS-PROTECTION-LIMIT               =        14
SYS-WORK-SPACE                     = 'EX0'
*****************************************************************************
TABLE &V0000 WAS DRIVEN BY PROGRAM PPTGGS
************************** TABLE SPECIFIC VARIABLES *********************
#1 = 'EX0      '
#2 = '                '
#3 = '20020122173850'
#4 = '0001'
#6 = 'MINERVA INTERNAL    '
#7 = '64979701649701440838 68  '
#8 = 'QUOTE'
#9 = '07.01.06 BUILD 03'
#A = '07.01.06 BUILD 01'
*****************************************************************************
```

EVALUATE structures

EVALUATE structures are used when the action to be taken depends on different variable values. The EVALUATE statement evaluates an expression and executes the statements matching the evaluation result.

Syntax:

> EVALUATE expression

[> WHEN expression

[> THRU expression]]°

[… instruction …]¹

[> WHEN OTHER

[… instruction …]°]

> END-EVALUATE

Where:

- **EVALUATE**:

  EVALUATE is the first MTL instruction of the EVALUATE-block. It is followed by an MTL expression. The content of this expression will be calculated. Subsequently, the result and the result type (numeric or alphanumeric) will be stored into memory.

- **expression**:

  expression refers to a regular MTL expression.

- **instruction**:

  instruction refers to COBOL code or an MTL instruction.

- **WHEN**:

  Several **WHEN** instructions can be put in an **EVALUATE** block. A **WHEN** instruction is followed by an MTL expression. The content of this expression will be calculated. Subsequently, the result and the result type (numeric or alphanumeric) will be compared with the result of the **EVALUATE** expression. If this result is the same, the instructions following the **WHEN** instruction will be carried out.

  Example:

```
> EVALUATE VAR1
> WHEN 'A'
*  'VAR1 IS 'A''
> WHEN 'C'
*  'VAR1 IS 'C''
> END-EVALUATE
```

- **THRU**:

  A **WHEN** instruction can be followed by a **THRU** instruction. That means that all expression values between the **WHEN** expression value on the previous line and the **THRU** expression value on this line will be compared with the **EVALUATE** expression value.

  Example:

```
> EVALUATE VAR1
>  WHEN 'A'
>  THRU 'C'
>   SET VAR2 = 'VAR1 IS 'A' OR 'B' OR 'C''
>  WHEN 'E'
>  THRU 'G'
>   SET VAR2 = 'VAR1 IS 'E' OR 'F' OR 'G''
> END-EVALUATE
```

- **Multiple WHEN**:

  Several **WHEN-THRU** instructions can be used one after another. In that case they will be combined.

  Example:

```
> EVALUATE VAR1
>  WHEN 'A'
>  WHEN 'B'
>   SET VAR2 = 'VAR1 IS 'A' OR 'B''
>  WHEN 'C'
>  WHEN 'G'
>  THRU 'K'
>   SET VAR2 = 'VAR1 IS IN 'CGHIJK''
> END-EVALUATE
```

- **WHEN OTHER**:

  WHEN OTHER is used to indicate all expression values that are not covered by previous **WHEN-THRU** expressions in current **EVALUATE** block.

  Example:

```
> EVALUATE VAR1
>  WHEN 'C'
>  WHEN 'E'
>  THRU 'H'
>   SET VAR2 = 'VAR1 IS IN 'CEFGH''
>  WHEN OTHER
>   SET VAR2 = 'VAR1 IS INVALID !'
> END-EVALUATE
```

- **END-EVALUATE**:

  This command terminates the **EVALUATE** instruction block.

Note: Nesting of multiple **IF** and **EVALUATE** blocks is possible. Up to 16 nesting levels are allowed.

EXIT table

The *EXIT* instruction allows exiting a table:

- in order to return to the calling table, if nesting level is not zero. For the EXIT statement the same rules are applied as for the GOTO statement.
- In order to proceed with the next table (if nesting level is zero).

Syntax:

>EXIT

Rule:

An EXIT can be done from inside an IF- or EVALUATE- block.

Fault Message handling

In the MetaSuite Template Language, you can program your own fault messages and stop the generating process, if required.

Four types of fault messages can be distinguished:

- *ERROR*:

  These are messages signalling serious errors that stop the generation process.

- *WARNING*:

  These are messages signalling less serious errors that do not stop the generation process.

- *MESSAGE*:

  These are messages informing the user of an event. The generation process will not be stopped.

- *USER*:

  These are informative messages defined by the user. They have nothing to do with the MESSAGE table in the MetaSuite dictionary.

Syntax:

>{ERROR|WARNING|MESSAGE} [message-number][message-text]

Where:

- **ERROR, WARNING** or **MESSAGE**:

  The required Fault Message type

- **message-number**:

  Enter the number matching the required message in the dictionary message table (AU0001) For more information about the generator errors, please refer to the section. If you omit this number, a default error number will be generated.

- **message-text**:

  You may enter a free text. If you don't, a default text will be displayed.

The following table gives an overview of the possible formats of the **ERROR** command and the resulting messages in the generated output:

Command	Generated output
>ERROR	**\TBG570E 1 GENERAL MTL MESSAGE (TABLE AV0000 LINE 0026)** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *E* = Error • *1* = sequential number of errors on this line • **GENERAL MTL ERROR** = Default error message • **TABLE AV0000 LINE 0026** = table and line where the error was invoked.
>ERROR message-text	**\TBG570E 1 GENERAL MTL ERROR TEST123** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *E* = Error • *1* = sequential number of errors on this line • **GENERAL MTL ERROR** = Default error message • **TEST123** = user-defined message text
>ERROR message-number	**\TBG570E 1 SPACE NOT ALLOWED (TABLE AV0000 LINE 0026)** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *E* = Error • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Error message matching the number, as available in the default message table **AU0001** • **TABLE AV0000 LINE 0026** = table and line where the error was invoked.
>ERROR message-number message-text	**\TBG570E 1 SPACE NOT ALLOWED TEST123** Where: • **TBG570** = MetaSuite Generator module generating the error • *E*= Error • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Error message matching the number, as available in the default message table **AU0001** • **TEST123** = user-defined message text

The following table gives an overview of the possible formats of the **WARNING** command and the resulting messages in the generated output:

Command	Generated output
>WARNING	**\TBG570W 1 GENERAL MTL MESSAGE (TABLE AV0000 LINE 0026)** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *W* = Warning • *1* = sequential number of errors on this line • **GENERAL MTL MESSAGE** = Default warning message • **TABLE AV0000 LINE 0026** = table and line where the error was invoked.
>WARNING message-text	**\TBG570W 1 GENERAL MTL MESSAGE TEST123** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *W* = Warning • *1* = sequential number of errors on this line • **GENERAL MTL MESSAGE** = Default warning message • **TEST123** = user-defined message text
>WARNING message-number	**\TBG570W 1 SPACE NOT ALLOWED (TABLE AV0000 LINE 0026)** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *W* = Warning • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Warning message matching the number, as available in the default message table **AU0001** • **TABLE AV0000 LINE 0026** = table and line where the error was invoked.
>WARNING message-number message-text	**\TBG570W 1 SPACE NOT ALLOWED TEST123** Where: • **TBG** = MetaSuite Generator module generating the error • **570** = error number • *W* = Warning • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Warning message matching the number, as available in the default message table **AU0001** • **TEST123** = user-defined message text

The following table gives an overview of the possible formats of the **MESSAGE** command and the resulting messages in the generated output:

Command	Generated output
>MESSAGE	\
>MESSAGE message-text	**\THIS IS A USER MESSAGE** Where: • **THIS IS A USER MESSAGE** = the user-defined message-text

Command	Generated output
>MESSAGE message-number	**\TBG125 1 SPACE NOT ALLOWED (TABLE AV0000 LINE 0026)** Where: • **125** = the indicated message-number • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Message matching the message number, as available in the default message table **AU0001**
>MESSAGE message-number message-text	**\TBG125 1 SPACE NOT ALLOWED TEST123** Where: • **125** = the indicated message-number • *1* = sequential number of errors on this line • **SPACE NOT ALLOWED** = Warning message matching the number, as available in the default message table**AU0001** • **TEST123** = user-defined message text

FOR-NEXT loops

FOR-NEXT loops are used to perform structured loops. They help the table programmer to program loops more easily.

Syntax:

> FOR variable-1 = value-1 TO value-2 [STEP value-3]

[… instruction …][1]

> NEXT [variable]

Where:

- **value**:

  All values in the **FOR** instruction must be numeric.

- **instruction**:

  The instruction may be COBOL code or an MTL instruction.

- **FOR**:

  The first part is the variable name (**variable-1**) that will vary each time the FOR-loop is repeated.

  The *variable-1* name is followed by the = sign, which serves as a delimiter.

  The second part, **value-1** must be numeric. It is the first value the **FOR-variable** will get.

  value-1 is followed by the word **TO**, which must be used as delimiter.

  value-2 must be numeric as well. It is the highest value the **FOR-variable** may obtain. If the value of **variable-1** is not within the range from **value-1** to **value-2**, the **FOR** loop will end.

- **STEP**:

  The **STEP** value is the value by which the variable will be changed each time the FOR-NEXT loop is carried out. The default **STEP** value is one. You can change this by specifying another value after the word STEP (value-3). The STEP value can be negative.

- **NEXT**:

  The **NEXT** instruction indicates the end of the *FOR* loop. All text after the **NEXT** command is purely informational.

Note: Nesting of multiple FOR-loops is possible. Up to 16 nesting levels are allowed. The number of loops must be at least one! If not, an error message will be displayed.

Example:

```
>FOR I = 1 TO 4
>SET PRODUCT=0
>FOR J = 1 TO 5
>SET PRODUCT=[PRODUCT]+[I]
>SET TABLE([I].[J])=PRODUCT
>IF PRODUCT >= 10
* [I]*[J] IS AT LEAST 10
>END-IF
>NEXT J
>NEXT I
```

Results:

```
* 2*5 IS AT LEAST 10
* 3*4 IS AT LEAST 10
* 3*5 IS AT LEAST 10
* 4*3 IS AT LEAST 10
* 4*4 IS AT LEAST 10
* 4*5 IS AT LEAST 10
```

FREEZE and UNFREEZE

The *FREEZE* command stops the code table interpreter (of which the MTL interpreter is a part) from:

- changing the quotes in the code table to the standard quote setting

  Syntax:>**FREEZE QUOTE**
- changing table specific variables (#1, #2, #A, etc…), so that code table developers can leave the "#" characters as they are.

  Syntax:>**FREEZE TS-VARS**
- changing MTL variables, so that code table developers can leave the "[" and "]" characters as they are.

  Syntax:>**FREEZE MTL-VARS**
- changing the indent in the code table to the standard indent setting

  The indent of the generated code will be automatically defined by the position of the text in the table, and also by the position of the CALL statement in case of a called table. The FREEZE command stops this standard behavior and will use the standard indent settings of the table.

  Syntax:>**FREEZE INDENT**

UNFREEZE restarts the normal code line treatment:

Syntax:

- **>UNFREEZE QUOTE**
- **>UNFREEZE TS-VARS**
- **>UNFREEZE MTL-VARS**
- **>UNFREEZE INDENT**

Examples:

```
BMOVE 'NO FREEZE QUOTE STATE' TO SENTENCE.
>FREEZE QUOTE
BMOVE 'FREEZE QUOTE STATE' TO SENTENCE.
>UNFREEZE QUOTE
BMOVE 'AGAIN NO FREEZE QUOTE STATE' TO SENTENCE.
```

This results in the following generated code:

```
      MOVE 'NO FREEZE QUOTE STATE' TO SENTENCE.
      MOVE 'FREEZE QUOTE STATE' TO SENTENCE.
      MOVE 'AGAIN NO FREEZE QUOTE STATE' TO SENTENCE.
```

GOTO

GOTO is a jump command that allows skipping or repeating certain MTL commands and COBOL code in the generated MGL.

Syntax:

>GOTO { label | line number}

Where:

- **GOTO**:

  The **GOTO** statement is used for jumping to another line WITHIN the current table. First of all the Generator will search for the label or line number in the lines following the current line. If the label or line number is not found, the table will be read again and the search will continue. If the label or line number is not found the second time, an error will be generated.

- **label**:

  Labels are indicators marking a specific point in a table. A label can be any kind of MTL instruction that is not a known command. It can contain one to eight characters. If a label contains more than eight characters, only the first eight characters will be taken in account. Labels can replace line numbers in the jump instructions.

- **line number**:

  Line numbers are represented using 4 digits from 0001 to 9999. You can find out the line number of a certain instruction in a table by executing the *LIST TABLE* instruction.

Restrictions:

The **GOTO** command can only be used:

- from inside an IF or EVALUATE block, to outside this block
- from inside a FOR-NEXT loop towards that same FOR-NEXT loop

A jump can never be performed from outside a structure towards inside that structure.

Example:

```
>REMARK A GOTO CAN BE DONE UPWARDS
>REMARK BUT SEARCHES THE LABEL WITHIN THE TABLE
>GOTO LABEL2
>
* THIS LINE WILL BE SKIPPED
>LABEL2
* THIS IS THE LINE AFTER THE LABEL2 LABEL
*
* LOOP EXAMPLE *
>SET NUMBER=10
&01 TABLE-X.
B02 TABLE-X-ELEMENT OCCURS [NUMBER] TIMES
B                   PIC X(15).
&01 REDEFINES TABLE-X.
>SET COUNTER=0
>LABEL0
>IF COUNTER <> [NUMBER]
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS  '[COUNTER]' *
>       SET COUNTER=[COUNTER]+1
B 02 TABLE-ELEMENT-[COUNTER] PIC X(15).
> GOTO LABEL0
>END-IF
```

Result:

```
* THIS IS THE LINE AFTER THE LABEL2 LABEL
*
* LOOP EXAMPLE *
01 TABLE-X.
        02 TABLE-X-ELEMENT OCCURS 10 TIMES PIC X(15).
01 REDEFINES TABLE-X.
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '0' *
        02 TABLE-ELEMENT-1 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '1' *
        02 TABLE-ELEMENT-2 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '2' *
        02 TABLE-ELEMENT-3 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '3' *
        02 TABLE-ELEMENT-4 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '4' *
        02 TABLE-ELEMENT-5 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '5' *
        02 TABLE-ELEMENT-6 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '6' *
        02 TABLE-ELEMENT-7 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '7' *
        02 TABLE-ELEMENT-8 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '8' *
        02 TABLE-ELEMENT-9 PIC X(15).
* COUNTER IS NOT EQUAL TO 10 BECAUSE IT IS '9' *
        02 TABLE-ELEMENT-10 PIC X(15).
```

IF structures

IF structures are necessary to perform condition-dependent actions.

Syntax:

>IF expression compare-operator expression

[{ >OR expression compare-operator expression

| >AND expression compare-operator expression }]°

[> THEN] [… MTL-instruction …]

[… instruction …]°

[> ELSE [… MTL-instruction …]

[… instruction …]°]

> END-IF

Where:

- **compare-operator** is one of the following:

  = equals

  <> differs from

  < is less than

  > is more than

  >= is not less than

  <= is not more than

- **expression** is a regular MTL expression
- **instruction** is a piece of COBOL code or an MTL instruction
- **MTL-instruction** is an MTL instruction or no instruction
- **IF/AND/OR** are Boolean operators used to define the condition for which certain actions have to be taken. Multiple AND-OR combinations may follow the IF instruction.

  AND is stronger than *OR*:

  IF condition1 AND condition2 OR condition3 means that both condition1 and condition2 must be met, or if this is not the case, that condition3 has to be met.

  Example:

```
> IF MIN < MAX
> AND X > [MIN] + 1
> AND X < [MAX] - 1
> THEN SET CON = 'X IS STRICTLY BETWEEN [MIN] AND [MAX]'
B SUBTRACT [X] FROM F#1-R#1-RECORD-SIZE.
> ELSE SET CON = 'X IS NOT STRICTLY BETWEEN [MIN] AND [MAX]'
B ADD 1 TO F#1-R#1-RECORD-SIZE.
> END-IF
```

- **THEN**

  The **THEN** command follows the IF-AND-OR combination. It can be followed by an MTL command on the same line. The lines after the **THEN** command can also be MTL commands, or might also be COBOL lines that have to be written down in the generated COBOL program.

  All instructions between **THEN** and **END-IF**, or between **THEN** and **ELSE** will be carried out in case the **IF-AND-OR** combination is true.

- **ELSE**

  The **ELSE** command is the opposite of the **THEN** command: the instructions between **ELSE** and **END-IF** will be carried out in case the **IF-AND-OR** combination is false. **ELSE** can be followed by an MTL command on the same line.

- **END-IF**

  END-IF terminates the IF instruction block.

IMPLEMENT

The *IMPLEMENT* command is used to call a section at a later stage. It remembers which tables are to be called, and calls them at the end of the MGL generation.

Syntax:

>IMPLEMENT <tablename> [param1] [param2] [param3] [param4]

Result:

The table `<tablename>` will be generated at the end of the MGL, using the parameters [param1] [param2] [param3] [param4].

Within the table, the parameters can be used as #1, #2, #3 and #4.

NESTED IF

It is possible to nest multiple **IF** and **EVALUATE** blocks. Up to 16 nesting levels are allowed.

Example:

```
>SET  X=5
>SET  Y=6
>IF X > Y
>THEN SET M='X IS BIGGER THAN Y'
*                    X>Y       *
>ELSE IF X = Y
>      THEN SET M='X EQUALS Y'
*                    X=Y       *
>      ELSE SET M='Y IS BIGGER THAN X'
*                    X<Y       *
>      END-IF
>END-IF
* MESSAGE : [M] *
```

Result:

```
*                    X<Y       *
* MESSAGE : Y IS BIGGER THAN X *
```

*** X<Y ***

*** MESSAGE: Y IS MORE THAN X ***

Since X is set to 5 and Y is set to 6, the first **IF** instruction will be false.

The **ELSE** expression will be carried out, and the second **IF** instruction will be executed. Since X is not equal to Y, the second expression will be false as well. The second **ELSE** instruction will be taken and the variable **M** will become **Y IS BIGGER THAN X**.

REMARK

The *REMARK* command has an informational purpose. It provides the user with the possibility of putting some comment in the MTL code.

Syntax:

>REMARK THIS LINE WILL BE SHOWN WHEN TRACE IS ON (LOW)

or

>* THIS LINE WILL BE SHOWN WHEN TRACE IS ON (HIGH)

Result:

If you generate this in TRACE-ON mode, the remark lines can be seen in comment mode.

```
>REMARK A SKIP CAN BE DONE DOWNWARDS BUT NOT UPWARDS
>REMARK SKIP IS THE ONLY STRUCTURE THAT CAN PASS
>REMARK THE BORDER OF A TABLE
>SKIP LABEL1
>REMARK THIS LINE WILL BE SKIPPED
```

SET

The *SET* command creates or modifies an MTL variable.

Syntax:

>SET variable-name = expression variable-name

Where:

- **variable-name** is the name of the variable that will be declared and will obtain a certain value by the SET command. It will be called the "SET-variable".
- **expression** is the expression that will be investigated. The resulting value will become the value of the SET-variable.

Example:

```
*************************************************
**** WORKING WITH CHARACTER VARIABLES  ****
*************************************************
*
>SET  &V0000-USER-NAME='#6'
>SET  &V0000-PROGRAM-ID='#1'
>SET  MESSAGE1='#6 HAS WRITTEN #1 ON #3'
>SET  MESSAGE2='[&V0000-USER-NAME] HAS WRITTEN '
>SET  MESSAGE2='[MESSAGE2][&V0000-PROGRAM-ID] ON '
>SET  MESSAGE2='[MESSAGE2][SYS-DATE-WRITTEN]'

* MESSAGE1 = '[MESSAGE1]'
* MESSAGE2 = '[MESSAGE2]'

>SET  WOORDQ='AZERTY''BZERTY'
>SET  WOORDDQ='AZERTY''BZERTY'

* WOORDQ = [[WOORDQ]] & WOORDDQ = [[WOORDDQ]] *
*
*******************************************************
*** WORKING WITH NUMERIC VARIABLES.                 *
*** SINCE NUMERIC EXPRESSIONS CAN ONLY CONTAIN      *
*** NUMERIC OPERATORS AND NUMERIC CHARACTERS.       *
*** WE'LL HAVE TO WORK WITH THE VARIABLE CONTENTS *
*******************************************************
*
>SET  X=5
>SET  Y=X
>SET  X=[X]+4
>SET  Y=[Y]-2*[X]
>SET  Z=[X]+10*[Y]/[X]
* X=[X], Y=[Y], Z=[Z] *
```

Result:

```
*************************************************
**** WORKING WITH CHARACTER VARIABLES  ****
*************************************************
*
* MESSAGE1 = '[MESSAGE1]'
* MESSAGE2 = '[MESSAGE2]'
* WOORDQ =  [AZERTY'BZERTY] & WOORDDQ = [AZERTY'BZERTY] *
*
*******************************************************
*** WORKING WITH NUMERIC VARIABLES.                 *
*** SINCE NUMERIC EXPRESSIONS CAN ONLY CONTAIN      *
*** NUMERIC OPERATORS AND NUMERIC CHARACTERS.       *
*** WE'LL HAVE TO WORK WITH THE VARIABLE CONTENTS *
*******************************************************
*
* X=9, Y=-13, Z=-5 *
```

SKIP

SKIP is a jump command allowing to skip or repeat certain MTL commands and COBOL code in the generated MGL.

Syntax:

>SKIP { label | line number }

Where:

- **SKIP**:

  The **SKIP** statement is used for jumping to another line **after** the current line of the current table. First of all the Generator will search for the label or line number in the lines after the current line. If the label or line number is not found, the next table will be read, and the search will continue. If the label or line number is not found after reading and searching in all subsequent tables, an error will be generated. A **SKIP** statement **can go beyond a single table**!

- **label**:

  Labels are indicators marking a specific point in a table. A label can be any kind of MTL instruction that is not a known command. It can contain one to eight characters. If a label contains more than eight characters, only the first eight characters will be taken in account. Labels can replace line numbers in the jump instructions.

- **line number**:

  Line numbers are represented using 4 digits from 0001 to 9999. You can find out the line number of a certain instruction in a table by executing the *LIST TABLE* instruction.

Example:

```
>REMARK A SKIP CAN BE DONE DOWNWARDS BUT NOT UPWARDS
>REMARK SKIP IS THE ONLY STRUCTURE THAT CAN PASS
>REMARK THE BORDER OF A TABLE
>SKIP LABEL1
* THIS LINE WILL BE SKIPPED
>LABEL1
* THIS IS THE LINE AFTER THE LABEL1 LABEL
* THIS COMMAND WILL SKIP THE REST OF CURRENT TABLE
>SKIP 0001
* THIS LINE WILL NOT BE GENERATED
```

Results:

```
* THIS IS THE LINE AFTER THE LABEL1 LABEL
* THIS COMMAND WILL SKIP THE REST OF CURRENT TABLE
```

TRACE

The *TRACE* command is used for tracing or debugging the MTL code. It provides more information about the logic followed by the MTL interpreter.

You can use the TRACE command with the following options:

- **OFF**: no MTL sentences will appear in the generated MGL/MRL.
- **ON** or **LOW**: only the relevant MTL statements will be shown as comment.

  For example: in an IF-THEN-ELSE structure only the branch that was followed (**IF or ELSE**) will be shown in the generated MGL or MRL.

- **MEDIUM**: Conditional branches will be shown in comment, but skipped MTL statements due to jump instructions (GOTO and SKIP) will not be shown.
- **HIGH**: Every MTL line will be put as comment in the MGL or MRL, including the lines that are skipped due to a jump instruction. Even the remarks that begin with ">*" will be put in the MGL or MRL.

Syntax:

>TRACE { OFF | ON | LOW | MEDIUM | HIGH }

Examples:

Sample code:

```
>TRACE LOW
>SET X=5
>IF X=3
>  REMARK X IS THREE
>ELSE
>  REMARK X IS SOMETHING ELSE THAN THREE
>  GOTO THE-END
>END-IF
>REMARK NOT THE END
>THE-END
>REMARK THIS IS THE END
```

Result:

When TRACE = LOW

```
      *TRACE LOW
      *SET X=5
01 T  *ELSE
01 T  *  REMARK X IS SOMETHING ELSE THAN THREE
      *THE-END
      *REMARK THIS IS THE END
```

When TRACE = MEDIUM

```
      *TRACE MEDIUM

      *SET X=5
01 F  *IF X=3
01 F  *  REMARK X IS THREE
01 T  *ELSE
01 T  *  REMARK X IS SOMETHING ELSE THAN THREE
      *THE-END
      *REMARK THIS IS THE END
```

When TRACE = HIGH

```
      *TRACE HIGH
      *SET X=5
01 F  *IF X=3
01 F  *  REMARK X IS THREE
01 T  *ELSE
01 T  *  REMARK X IS SOMETHING ELSE THAN THREE
  G   *  GOTO THE-END
  G   *END-IF
  G   *REMARK NOT THE END
      *THE-END
      *REMARK THIS IS THE END
```

Remarks:

- The lines that were not interpreted in a conditional instruction are preceded by the letter *F* (for *False*) in column 4.
- The lines that were interpreted in a conditional instruction are preceded by the letter *T* (for *True*) in column 4.
- When interpreting conditional instructions, the nesting level is put in columns one and two.
- The lines that were not interpreted lines due to a jump instruction are preceded by the letter "G" in case of a GOTO, and by the letter "S" in case of a SKIP. Those letters are put in column three.

UNSET

The *UNSET* command removes an MTL variable.

Syntax:

>UNSET variable-name

Where:

- **variable-name** is the name of the variable that will be removed.

Example:

```
****************************************
**** WORKING WITH SET AND UNSET  ****
****************************************
>SET TEST=1
>SET X=FUNCTION IS-SET(TEST)
*TEST=[TEST] AND X=[X]
>UNSET TEST
>SET X=FUNCTION IS-SET(TEST)
*TEST=[TEST] AND X=[X]
```

Result:

```
**************************************
**** WORKING WITH SET AND UNSET  ****
**************************************
*TEST=1 AND X=TRUE
*TEST=0 AND X=FALSE
```

15.4. MTL System Variables

In the MetaSuite Template Language, some system variables have been predefined. These variables are called "protected" because they can not be changed by the SET command. They contain values defined in the MetaSuite dictionary.

The following table describes those MTL system variables.

Variable	Description	Example
SYS-CALL-DYNAMIC	This option is set to *Y*, when the generated COBOL uses dynamic COBOL calls to invoke the run-time modules. It is set to *N*, if the COBOL calls are not dynamic.	*Y* or *N*
SYS-COBOL	This is the COBOL compiler for which the MGL will be generated, represented as a 2-character string.	VS
SYS-COBOL-VERSION	This is the version of the COBOL compiler for which the MGL will be generated, represented as a 3-digit number.	085
SYS-DATE-FORMAT	This is the current date format setting.	*ISO*, *EUR* or *JIS*
SYS-DATE-WRITTEN	This is the generation date in the following format: *YYYYMMDD*.	20130411
SYS-DECIMAL-POINT	This is the current Decimal Point setting.	. or ,
SYS-EXEC	This is the current EXEC setting.	*IMS* or *NULL*
SYS-FIELD-COUNT	The number of fields in the "fields" table. This table contains target fields, work fields and source fields in this order.	156
SYS-FILE-COUNT	This is the number of source files.	20
SYS-GENERATOR-VERSION	This is the current Generator version, represented as a 5-digit number.	80103
SYS-MODEL-NAME	This is the name of the MetaSuite Model.	SampleModel
SYS-NULL	This is the current NULL character setting.	\
SYS-NULLABLE	This is the current nullable setting.	*N* or *I* (uppercase i)
SYS-OPERATING-SYSTEM	This is the operating system for which the MGL will be generated, represented as a 3-character string.	BS2
SYS-PROGRAM-NAME	This is the name of the generated program	SampleModel.exe

Variable	Description	Example
SYS-PROTECTION-LIMIT	This is the number of protected variables, including this one.	25
SYS-QUOTE	This is the current QUOTE setting.	' or "
SYS-RECORD-COUNT	This is the number of source records	24562
SYS-SQL-DIALECT	This is the current SQL dialect setting, represented as a value between 1 and 15.	0 = Not specified 1 = DB2 for z/OS 2 = DB2 for DOS/VSE 3 = DB2/2 4 = DB2 for OS/400 5 = DB2 LUW 6 = Oracle 7 = Ingres 8 = Sybase 9 = SQL Server 10 = Informix 11 = SESAM 12 = Oracle/RDB 13 = ODBC 14 = Teradata 15 = MySQL
SYS-USER	This is the name of the user who generated this program.	john
SYS-SQL-FOUND	If the generated MXL contains SQL related features, this variable gets the value "TRUE". If not: "FALSE".	
SYS-INTERPUNCTION	Contains the default INTERPUNCTION setting. It's value can either be "NO", "YES" or "OFF".	
SYS-TARGET-COUNT	The number of target files.	
SYS-RECORD-COUNT	The number of source records.	
SYS-DETAIL-COUNT	The number of target records.	
SYS-FIRST-FIELD	The symbolic number of the first source field in the fields table.	
SYS-UNICODE	If Unicode is supported at your site, the value will be "TRUE".	
SYS-ONE-CCSID	If only one single CCSID is used in the MXL file, this variable gets the value "TRUE". If not: "FALSE".	
DEFAULT-CCSID	This numeric variable contains the default CCSID for characters.	
UNICODE-CCSID	This numeric variable contains the default CCSID for Unicode.	
SYS-DEBUG	Is "TRUE" if the generation is performed with the DEBUG option.	

Variable	Description	Example
SYS-NATIONAL-IS-USED	If the MXL contains Unicode fields, this variable gets the value "TRUE". If not: "FALSE".	

15.5. SourceFile-specific MTL Variables

The MetaSuite Generator provides the MTL programmer with some file-specific variables. These variables are subscripted or indexed. The index is the internal file number. This index can have a value between 1 and SYS-FILE-COUNT.

The following table describes these MTL file-specific variables.

Variable	Description
FILE-DBNAME (...)	This is the database name.
FILE-LENGTH (...)	This is the file length. This length is the space that the generated COBOL program will take for defining the file control block. This is at least the size of the largest record in this file.
FILE-NAME (...)	This is the file name.
FILE-RECORD-FORMAT (...)	This is the record format as defined in the MetaStore database. Possible values are: • FIXED • VARIABLE • UNDEFINED
FILE-RECORDING-MODE (...)	This is the recording mode as specified in the MetaStore database. Possible values are: • ASCII • EBCDIC • NATIVE
FILE-TYPE (...)	This is the file type indicating whether a file is a parameter file, an external array with or without key, etc.
FILE-CONTAINS-UCD (...)	"TRUE" or "FALSE". "TRUE" if the file contains Unicode/National fields.
FILE-IS-DELIMITED (...)	"TRUE" or "FALSE". "TRUE" if the file is a delimited source file.
FILE-XPATH-COUNT (...)	The number of XPATH levels in the XPATH property of the source file. While issuing this command, the MTL XPATH pointer is set to the start of the XPATH of the source file. For instance: If the XPATH of file number 7 is '/XML/HEAD/CORNER' then FILE-XPATH-COUNT (7) will be 3.
FILE-XPATH-NEXT (...)	The next XPATH level in the XPATH property of the source file. For instance, in our previous example, FILE-XPATH-NEXT (7) will be 'XML'. The next time FILE-XPATH-NEXT (7) will be issued, the result will be 'HEAD', because this is the NEXT part of the XPATH. Each time this command is issued, the XPATH pointer moves forward, until it reaches the end of the XPATH string. At the end, the result will be an empty string.

Variable	Description
FILE-FIRST-FIELD (...)	The lowest symbolic field number of all fields that belong to the file. This function is developed in order to speed up the MTL code.
FILE-NBR-RECORDS (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.
FILE-IS-TERMINATED (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.
FILE-IS-OBF (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.
FILE-OBF-NUMBER (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.

15.6. SourceRecord-related MTL Variables

The MetaSuite Generator provides the MTL programmer with some record-specific variables. These variables are subscripted or indexed. The index is the internal record number. This index can have a value between 1 and SYS-RECORD-COUNT.

The following table describes these MTL record-specific variables.

Variable	Description
RECORD-DBNAME (...)	This is the database name.
RECORD-FILE (...)	This is the index of the source file this source record belongs to.
RECORD-LENGTH (...)	This is the record length.
RECORD-NAME (...)	This is the record name.
RECORD-OCCURRANCY (...)	This is the record occurrence number specified in the path.
RECORD-XPATH-COUNT (...)	The number of XPATH levels in the XPATH property of the source record. While issuing this command, the MTL XPATH pointer is set to the start of the XPATH of the record. If the XPATH of the source record is a relative path, then the XPATH of the file will be prefixed to this XPATH. For instance: Assume that record 10 is a record of file 7. If the XPATH of file number 7 is '/XML/HEAD/CORNER' and the XPATH of record number 10 is 'GOALS/POINTS', then RECORD-XPATH-COUNT (10) will be 5.
RECORD-XPATH-NEXT (...)	The next XPATH level in the XPATH property of the record. For instance, in our previous example, RECORD-XPATH-NEXT (10) will be 'XML'. The next time RECORD-XPATH-NEXT (10) will be issued, the result will be 'HEAD', because this is the NEXT part of the full XPATH. Each time this command is issued, the XPATH pointer moves forward, until it reaches the end of the XPATH string. At the end, the result will be an empty string.

Variable	Description
RECORD-RAW-NAME (...)	While RECORD-NAME contains only uppercase characters, RECORD-RAW-NAME contains the original record as was specified in the MXL.
RECORD-COLSEP (..)	Column separator of the source record.
RECORD-ROWTERM (..)	Row terminator of the source record.
RECORD-RCDCD-ELEM (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.
RECORD-RCDCD-ELEM-NXT (...)	This variable will not be supported anymore in future versions. Only to be used by MetaSuite Support.

15.7. TargetFile-related MTL Variables

The MetaSuite Generator provides the MTL programmer with some file-specific variables. These variablesare subscripted or indexed. The index is the internal target file number. This index can have a value between 1 and SYS-TARGET-COUNT.

The following table describes these variables:

Variable	Description
TARGET-XPATH-COUNT (...)	The number of XPATH levels in the XPATH property of the target file. While issuing this command, the MTL XPATH pointer is set to the start of the XPATH of the target file. For instance: If the XPATH of file number 7 is '/XML/HEAD/CORNER' then FILE-XPATH-COUNT (7) will be 3.
TARGET-XPATH-NEXT (...)	The next XPATH level in the XPATH property of the target file. For instance, in our previous example, FILE-XPATH-NEXT (7) will be 'XML'. The next time FILE-XPATH-NEXT (7) will be issued, the result will be 'HEAD', because this is the NEXT part of the XPATH. Each time this command is issued, the XPATH pointer moves forward, until it reaches the end of the XPATH string. At the end, the result will be an empty string.
TARGET-OUT-TYPE (...)	The type of target file. The result can be "T", "C", "D" for target files, and "R" or "P" for report files.
TARGET-OUT-NUMBER (...)	The external number that is used to refer to this target, both in the MXL as in the generated COBOL and scripts.
TARGET-IN-NUMBER (xx)	Only for non-report files: the internal symbolic number that is used for target file Txx. This is the reverse functionality of the previous function.
TARGET-XMLDECL-NBR (...)	An XML file often starts with an XML declaration sentence. This sentence can be a long string, which has to be cut into smaller parts in order to fit into a COBOL sentence. The number of parts in which this sentence will be cut, is the result of this functionality.

Variable	Description
TARGET-XMLDECL-NXT (...)	See also TARGET-XMLDECL-NBR. An XML file often starts with an XML declaration sentence. This sentence can be a long string, which has to be cut into smaller parts in order to fit into a COBOL sentence. This function returns a a part of the XML-declaration. For each new function call, a fresh part of the XML-declaration will be returned, until the end of the XML-declaration is reached. In that case, an empty value will be returned.
TARGET-CONTAINS-UNICODE (...)	If a target file contains Unicode fields, the value "TRUE" will be returned, else the value "FALSE" will be returned.

15.8. TargetRecord-related Variables

The MetaSuite Generator provides the MTL programmer with some file-specific variables.These variablesare subscripted or indexed. The index is the internal target record number, or "detail" number. This index can have a value between 1 and SYS-DETAIL-COUNT.Following table describes these MTL file-specific variables.

The following table describes these variables:

Variable	Description
DETAIL-NAME	The name of the target record.
DETAIL-TARGET	The symbolic number of the target file to which this record belongs to.
DETAIL-LINE	The detailed line number, specified in the MXL.
DETAIL-CCSID	The CCSID of the target record.
DETAIL-XPATH-COUNT	The number of XPATH levels in the XPATH property of the target record. While issuing this command, the MTL XPATH pointer is set to the start of the XPATH of the record. If the XPATH of the target record is a relative path, then the XPATH of the file will be prefixed to this XPATH. For instance: Assume that record 10 is a record of file 7. If the XPATH of file number 7 is '/XML/HEAD/CORNER' and the XPATH of record number 10 is 'GOALS/POINTS', then RECORD-XPATH-COUNT (10) will be 5.
DETAIL-XPATH-NEXT	The next XPATH level in the XPATH property of the target record. For instance, in our previous example, RECORD-XPATH-NEXT (10) will be 'XML'. The next time RECORD-XPATH-NEXT (10) will be issued, the result will be 'HEAD', because this is the NEXT part of the full XPATH. Each time this command is issued, the XPATH pointer moves forward, until it reaches the end of the XPATH string. At the end, the result will be an empty string.
DETAIL-TYPE	This variable can have 3 possible values: • "D" means DETAIL record • "T" means TOTAL (end-total record) • "A" means ACCUMULATE (group record)

15.9. Field-related MTL Variables

The MetaSuite Generator provides the MTL programmer with some field-specific variables. These variables are subscripted or indexed. The index is the internal record number. This index can have a value between 1 and SYS-FIELD-COUNT

The following table describes these MTL field-specific variables.

Variable	Description
FIELD-DATE-TYPE (...)	This is the date type expressed as a number. If set to 0, the field has no date type. The following Date Types exist: • 1 to 10 : NUMERIC types. That means that the numeric value is taken as input. • 11 to 30 : ALPHANUMERIC types. That means that the alpha value is taken as input. • 11 to 20 : ALPHANUMERIC with delimiter • 21 to 30 : ALPHANUMERIC without delimiter • The date format is determined by the last digit: - ending on 1 : MMDDYY - ending on 2 : DDMMYY - ending on 3 : MMDDYYYY - ending on 4 : DDMMYYYY - ending on 5 : YYDDD - ending on 6 : YYYYDDD - ending on 7 : YYMMDD - ending on 8 : YYDDMM - ending on 9 : YYYYMMDD - ending on 0 : YYYYDDMM
FIELD-DBNAME (...)	This is the DB-NAME setting of the field. It also may contain a NULL value.
FIELD-DECIMALS (...)	This is the number of decimals behind the decimal point.
FIELD-FILE (...)	This is the index of the source file this field belongs to. It is 0, if the field does not belong to a source file.
FIELD-GROUP (...)	This is the field number of the group field.
FIELD-SIZE-BYTES (...)	This is the number of bytes that are reserved for the field in the record.
FIELD-NAME (...)	This is the field name.
FIELD-OCCURS (...)	This is the total number of field occurrences of this field in one record. This number is obtained by multiplying all occurrences of current and higher grouping levels.
FIELD-OCCURS-LEVEL (...)	This is the occurrence value on the lowest level.
FIELD-ORIGIN (...)	This is the origin indication of the field: • G = group field • F = source field • W = work field • U = other field
FIELD-POSITION (...)	This is the absolute position of the field in the record.
FIELD-RAW-NAME (...)	This is the field name in the RAW format, including the *_nnnn#nnnnnn* suffix.

Variable	Description
FIELD-RECORD (...)	This is the index of the source record this field belongs to. The value is 0, if the field does not belong to a source file.
FIELD-SUBSCRIPT-LEVEL (...)	This is the number of hierarchical levels that have an occurrence of more than 1.
FIELD-TYPE (...)	This is the field type, expressed as a number.
FIELD-SEQUENCE (...)	Sequence number of the field. This variable will probably not be supported anymore in future versions of MetaSuite.
FIELD-MVOFLD (...)	Will not be supported anymore in future versions of MetaSuite. Only to be used by MetaSuite Support.
FIELD-DUMP (...)	Will not be supported anymore in future versions of MetaSuite.
FIELD-SIZE-BYTES (...)	The physical size of the field: the number of bytes.
FIELD-SIZE-CHARS (...)	The size of the field in number of characters.
FIELD-IS-CODESET-SENSITIVE (...)	Some field types are code set-sensitive, others are not. If this field is sensitive to code set altering, the returning value will be "TRUE", else it will be "FALSE".
FIELD-IS-NUMERIC (...)	If this is a numeric field, the returning value will be "TRUE", else it will be "FALSE".
FIELD-IS-SIGNED (...)	If this is a signed field, the returning value will be "TRUE", else it will be "FALSE".
FIELD-MULT (...)	Returns a value of 10 characters, containing following information: • The first byte contains "Y" if the field is multiple occurring • The second byte contains the number of occurring levels • Byte 3 to 6 : the number of occurrences on the first level • Byte 7 to 10 : the number of occurrences on the second level
FIELD-NRINDEX (...)	The number of indexes that can be used for this field.
FIELD-XMLNAME (...)	The XML name of the field. If the XMLNAME property is not empty, this value will be taken. If the XMLNAME property does not exits, the field name without prefix and without suffix will be taken.
FIELD-XMLTYPE (...)	The XML type of the field: attribute (value "A"), group field (value "G") or node (value space).
FIELD-XMLMASK (...)	The XML "edit" mask of field.
FIELD-IS-ATTRIBUTE (...)	Is "TRUE" when FIELD-XMLTYPE (...) = "A". Else it is "FALSE".
FIELD-HAS-ATTRIBUTES (...)	Is "TRUE" when this is a group field, and when at least one the subfields is an attribute. Else it is "FALSE".
FIELD-IS-GROUP (...)	Is "TRUE" when FIELD-XMLTYPE (...) = "G". Else it is "FALSE".
FIELD-IS-NULLABLE (...)	Is "TRUE" when the field is INNULL, OUTNULR, OUTNULL, or if it is DEFAULT and the DEFAULT-NULLABLE is "INNULL". In other words, this variable indicates whether the field is a nullable field or not.

Variable	Description
FIELD-INIT (...)	If a field has an initial value, this function returns the value "TRUE", else the result is "DEFAULT". Exception: if a field has a group field, and this group field has an initial value, then the group field value is has priority on the subfield value. In that case the function returns "FALSE".
FIELD-CCSID (...)	The CCSID on field level. CCSID is the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. This CCSID can be specified on general MetaSuite level (dictionary settings CHARACTER-CCSID and UNICODE-CCSID), but also on file level, record level and field level.
FIELD-CCSID-TYPE (...)	The FIELD-CCSID-TYPE is 1 for character CCSIDs, and 2 for Unicode CCSIDs.
FIELD-CCSID-LEN1 (...)	The number of characters that match this field.
FIELD-CCSID-LEN2 (...)	The size a Unicode field should have if this field would be moved to a Unicode field. (In most cases: FIELD-CCSID-LEN1 (...) * 2)
FIELD-IS-TYPNAT (...)	Is "TRUE" if the field type is "NATIONAL".
FIELD-SUBSCRIPT-LEVEL (...)	The number of subscripts that this field requires.
FIELD-SUBSCRIPT-HOLDER (...)	For elementary or one-dimensional fields: the group that holds the index.
FIELD-GROUP (...)	Returns the symbolic number of the group field, or zero if this is an elementary field.
FIELD-IS-1ST-KEY (...)	Returns "TRUE" if this field is the first key field.

CHAPTER 16
MetaSuite Run Language

The MetaSuite Run Language (MRL) is a series of commands or statements you can use to run your programs.
MRL statements are created by the Generator utility based on template tables.

16.1. Tables

The following tables are used by the MetaSuite Generator to create the MRL.

RL0000	Script header: #1 Program name #2 Program version number This table can be used to delete PPTLST and PPTLOG from a previous run.
RL0001	Program execution: #1 Program name #2 Program version number This table contains the command to run the program.
RL0002	External array Source File assignment (PPTFnn): #1 Program name #2 MetaSuite external array number #3 MetaSuite external array name
RL0003	SourceFile assignment (PPTFnn): #1 Program name #2 MetaSuite SourceFile number #3 MetaSuite SourceFile name
RL0004	Temporary file assignment (PPTTnn): #1 Program name #2 MetaSuite Source File number for which temporary file is needed (due to a SourceFile Sort or a SourceFile Extract)
RL0005	Data Target File assignment (PPTTDnn): #1 Program name #2 MetaSuite TargetFile number #3 Record length #4 Output file sequence number

RL0006	Format Target File assignment (PPTTFnn): #1 Program name #2 MetaSuite TargetFile number #3 Record length #4 Output file sequence number This table has become obsolete from V6.03 onwards.
RL0007	Report file assignment (PPTRnn): #1 Program name #2 MetaSuite TargetFile (report) number #4 Output file sequence number
RL0009	Target sort file assignment: #1 Program name
RL0010	INI file assignment (PPTIPT): #1 Program name #2 Input file sequence number
RL0011	LST file assignment (PPTLST): #1 Program name #2 Output file sequence number
RL0012	LOG file assignment (PPTLOG): #1 Program name #2 Output file sequence number
RL0013	Input network DBMS specific file assignment: IDMS
RL0014	Input relational DBMS specific file assignment: DB2, SESAM
RL0015	Input hierarchical DBMS specific file assignment: IMS
RL0016	Input ADABAS/C specific file assignment
RL0017	Relational program execution: DB2, SESAM #1 Program name #2 Program version number
RL0018	ADABAS/C program execution: #1 Program name #2 Program version number
RL0019	Hierarchical program execution: IMS #1 Program name #2 Program version number
RL0020	Network program execution: IDMS #1 Program name #2 Program version number
RL0021	Output file scratch for PPTTFnn: #1 Program name #2 MetaSuite TargetFile number (This table has become obsolete from MetaSuite V6.03 onwards.)
RL0022	Data Target File scratch for PPTTDnn: #1 Program name #2 MetaSuite TargetFile number
RL0024	Output report scratch for PPTRnn: #1 Program name #2 MetaSuite TargetFile (report) number

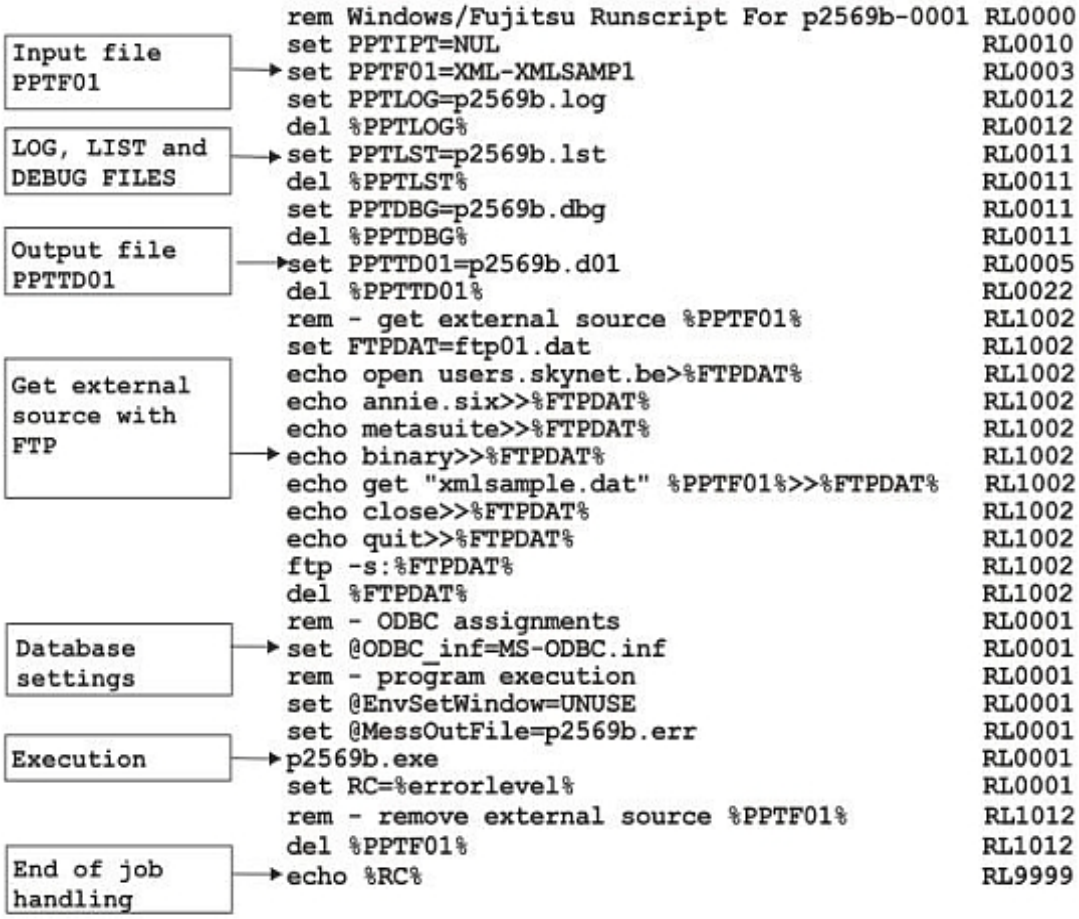
RL0025	Temporary file scratch for PPTTnn: #1 Program name #2 MetaSuite Source File number for which temporary file is needed (due to a SourceFile Sort or a SourceFile Extract)
RL0026	Controlled output file assignment: #1 Program name #2 MetaSuite controlled Source File number
RL0029	Remove target sort workfiles: #1 Program name #2 MetaSuite TargetFile number
RL0032	Output file scratch for controlled output file PPTTDnn: #1 Program name #2 MetaSuite TargetFile number
RL0100	Creation of parameter file - part 1
RL0101	Parameter default assignment for alphanumeric parameter fields
RL0102	Parameter default assignment for numeric parameter fields
RL0103	Parameter default assignment for date parameter fields
RL0199	Creation of parameter file - part 2
RL1002	Get or download external source file #2 Input file sequence number #3 Local file name, input file for the generated executable #5 External source type This contains the word "STD" if no colon was found on position 4 of the EXTERNAL SOURCE parameter. If a colon was found on position 4 of the EXTERNAL-SOURCE parameter, the value before the colon is put in #5. #6 External source file, remote file name
RL1003	VSAM SourceFile assignment (PPTFnn): #1 Program name #2 MetaSuite SourceFile number #3 MetaSuite SourceFile name
RL1010	LID file assignment (PPTLID): #1 Program name #2 Input file sequence number
RL1012	Remove downloaded external source file #2 Input file sequence number #3 Local file name, input file for the generated executable - this is the file that can be removed. #5 External source type This contains the word "STD" if no colon was found on position 4 of the EXTERNAL SOURCE parameter. If a colon was found on position 4 of the EXTERNAL-SOURCE parameter, the value before the colon is put in #5. #6 External source file, remote file name
RL9999	Script trailer: #1 Program name #2 Program version number

16.2. Examples

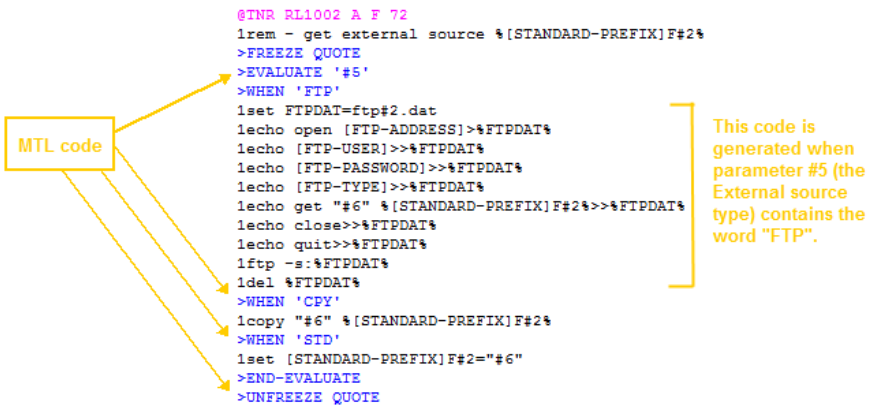
Some examples are listed here, in order to give an idea of what is possible.

Windows

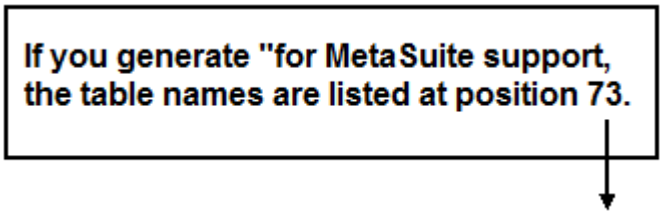
This example makes use of the "EXTERNAL SOURCE" facility.



If we look at the definition of table RL1002, we can see that this table contains a mix of table specific parameters (#2, #5, #6…), MTL parameters ([FTP-ADDRESS]…) and job parameters (%PATH%, %PPTF#2%…)

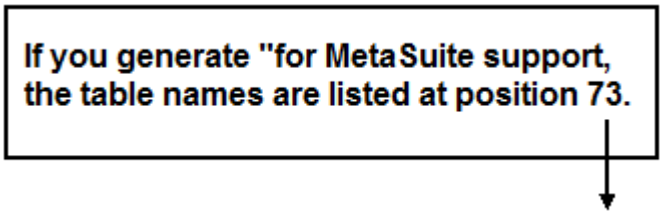


UNIX / Linux




```
# Unix/MicroFocus Runscript For c00-0001                RL0000
export dd_PPTIPT=NUL                                    RL0010
export dd_PPTT01=c00.t01                                RL0004
export dd_PPTLOG=c00.log                                RL0012
rm -f $dd_PPTLOG                                        RL0012
export dd_PPTLST=c00.lst                                RL0011
rm -f $dd_PPTLST                                        RL0011
export dd_PPTTD01=c00.d01                               RL0005
export dd_PPTR01=c00.R01                                RL0007
export dd_PPTTD02=c00.d02                               RL0005
rm -f $dd_PPTTD01                                       RL0022
rm -f $dd_PPTR01                                        RL0024
rm -f $dd_PPTTD02                                       RL0022
rm -f $dd_PPTT01                                        RL0025
export dd_PPTDBG=c00.dbg                                RL0001
# export TMPDIR=sortwork-temp-dir                       RL0001
# export COBSW=+D+S5-s-F                                RL0001
# export COBEXTFHBUF=nnnnnn                             RL0001
c00                                                     RL0001
rc=$?                                                   RL0001
export rc                                               RL0001
rm -f $dd_PPTT01                                        RL0025
echo "returncode is '$rc'"                              RL9999
```

VMS




```
$! VAX Runscript For ex0-0004                          RL0000
$DEL ex0.LOG                                           RL0000
$DEL ex0.LST                                           RL0000
$DEFINE PPT$IPT                                        RL0010
$DEFINE PPT$F01 employee-master                        RL0003
$DEFINE PPT$LOG ex0.log                                RL0012
$DEFINE PPT$LST ex0.lst                                RL0011
$DEFINE PPT$TD01 ex0.d01                               RL0005
$DEL ex0.d01                                           RL0022
$RUN ex0.EXE                                           RL0001
$!                                                     RL9999
```

OSD/BC BS2000

```
/LOGON
/SYSFILE SYSDTA=(SYSCMD)
/OPTION DUMP=YES,MSG=FHL
/CREATE-JV JV-NAME=JV-PARM
/SET JV-LINK JV-NAME=JV-PARM
/MOD-JV JV-PARM,'..parm ..'
/SYSFILE SYSLST=EX0.LST
/ERASE EX0.LOG
/ERASE EX0.LST
/ERASE EX0.D01
/ADFL LINK=PPTF01,F-NAME=employee-master
/CRF F-NAME=EX0.D01,SUP=*PUB-DISK(VOL=*STD, -
/    SPACE=*REL(PRIM-ALLOC=32,SEC-ALLOC=16))
/ADFL LINK=PPTTD01,F-NAME=EX0.D01,BUF-LEN=*STD(SIZE=16)
/ADFL LINK=PPTIPT,F-NAME=*DUMMY
/CRF F-NAME=EX0.LOG
/ADFL LINK=PPTLOG,F-NAME=EX0.LOG
/REMARK
/ADFL LINK=COBOBJCT,F-NAME=**LOADLIB**
/ADFL LINK=BLSLIB00,F-NAME=$TSOS.SYSLNK.CRTE
/SYSFILE SYSDTA=(SYSCMD)
/START-PROGRAM FROM-FILE=-
/      *MODULE( -
/               LIBRARY=**LOADLIB** -
/              ,ELEMENT=EX0 -
/              ,PROGRAM-MODE=ANY -
/              ,RUN-MODE=ADVANCED -
/              (ALTERNATE-LIBRARIES=YES -
/              ,UNRES-EXT=DELAY -
/              ,LOAD-INFO=REFERENCE ) -
/             )
/LOGOFF
```

OS/390 Z/OS

```
//C00 JOB **ACC**,'**ID**',MSGCLASS=**MCLS**,
//         MSGLEVEL=**MLVL**,REGION=4096K,
//         CLASS=**JCLS**,NOTIFY=**USER**
//DEL      EXEC PGM=IEFBR14
//LOG      DD DSN=C00.LOG,DISP=(MOD,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//D01      DD DSN=C00.D01,DISP=(MOD,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//*
//D02      DD DSN=C00.D02,DISP=(MOD,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//T01      DD DSN=C00.T01,DISP=(MOD,DELETE,DELETE),
//            SPACE=(TRK,(1,1))
//PARM     EXEC PGM=IEBGENER,COND=(0,NE)
//SYSPRINT DD   SYSOUT=**PCLS**
//SYSIN    DD   DUMMY
//SYSUT1   DD   *
*SYS-NUMERIC-CHECK = IGNORE
*SYS-DATE-CHECK = IGNORE
/*
```

```
//SYSUT2   DD   UNIT=**TEMPDA**,
//              DISP(,PASS,DELETE),
//              DCB=(RECFM=FBA,LRECL=255,BLKSIZE=27795),
//              SPACE=(TRK,(1,1),RLSE)
//RUN       EXEC PGM=IKJEFT01,COND=(0,NE),DYNAMNBR=20,REGION=**DB2REGN*
//STEPLIB   DD DSN=**DB2LIB**,DISP=SHR
//          DD DSN=**QUAL**.USER.LOADLIB,DISP=SHR
//          DD DSN=**SRTLIB**,DISP=SHR
//          DD DSN=**CBLLIB**,DISP=SHR
//          DD DSN=**QUAL**.LOADLIB,DISP=SHR
//SYSPROC   DD DSN=**DB2CLIB**,DISP=SHR
//DBRMLIB   DD DSN=**QUAL**.DBRM.DB2,DISP=SHR
//SYSDBOUT  DD SYSOUT=**PCLS**
//SYSPRINT  DD SYSOUT=**PCLS**
//SYSOUT    DD SYSOUT=**PCLS**
//SYSTSPRT  DD SYSOUT=**PCLS**
//SYSTERM   DD SYSOUT=**PCLS**
//SYSIN     DD DUMMY
//SORTLIB  DD DSN=**SRTLIB**,DISP=SHR
//SORTWK01 DD UNIT=**TEMPDA**,SPACE=(**SRTTRK**,,CONTIG)
//SORTWK02 DD UNIT=**TEMPDA**,SPACE=(**SRTTRK**,,CONTIG)
//SORTWK03 DD UNIT=**TEMPDA**,SPACE=(**SRTTRK**,,CONTIG)
//SORTWK04 DD UNIT=**TEMPDA**,SPACE=(**SRTTRK**,,CONTIG)
//PPTIPT   DD DSN=*.PARM.SYSUT2,
//            DSP=SHR
//PPTLOG   DD DSN=C00.LOG,DISP=(NEW,CATLG),DCB=(RECFM=V),
//         SPACE=(TRK,(1,1),RLSE),UNIT=SYSDA
//PPTLST   DD SYSOUT=**PCLS**
//PPTDBG   DD SYSOUT=**PCLS**
//PPTT01   DD DSN=C00.T01,DISP=(NEW,PASS),
//         SPACE=(CYL,(10,2))
//PPTTD01  DD  DSN=C00.D01,DISP=(NEW,CATLG),
//         SPACE=(TRK,(1,1),RLSE),UNIT=SYSDA
//PPTR01   DD SYSOUT=**PCLS**
//PPTTD02  DD  DSN=C00.D02,DISP=(NEW,CATLG),
//         SPACE=(TRK,(1,1),RLSE),UNIT=SYSDA
//SYSTSIN  DD *
PROFILE PREFIX(**DB2PREF**)
DSN SYS(**DB2SSID**)
BIND PLAN (C00) -
 MEMBER (C00) -
 QUALIFIER(**DB2QUAL**) -
 ACTION (REPLACE) ISOLATION (CS)
RUN PROGRAM (C00) PLAN (C00) -
 LIB ('**QUAL**.USER.LOADLIB') -
 PARM ('/')
//*
```

CHAPTER 17
String Modules

In order to facilitate string manipulations from within a MetaSuite program, a number of subroutines, called String Modules, can be made available to the programmer.

In total five modules have been developed: one for initialization purposes, and four for actual string manipulations.

17.1. Description

INITSTR	Initializes all parameter fields. See INITSTR - Initialize parameter fields on page 135.
LENSTR	Determines the length of a string, ignoring trailing spaces. See LENSTR - Determine the length of a string on page 136.
SUBSTR	Returns a substring from a given (input-)string. See SUBSTR - Return a string from a string on page 137.
SRCHSTR	Searches for occurrences of a search argument in a string. See SRCHSTR - Search in a string on page 139.
REPLSTR	Replaces occurrences of a (search-)argument in a string by a replace string. See REPLSTR - Replace substring in a string on page 141.

17.2. Usage

The following dictionary file (MDL) should be available in the MetaStore:

```
DELETE FILE FunctionString
ADD FILE FunctionString TYPE FUNCTION FIXED 1038
RULE MetaSuite String Functions.
ADD RECORD String-Parms SIZE 1038
ADD FIELD String-In             POSITION    1 SIZE 256 TYPE CHARACTER
ADD FIELD String-Out            POSITION  257 SIZE 256 TYPE CHARACTER
ADD FIELD String-Search         POSITION  513 SIZE 256 TYPE CHARACTER
ADD FIELD String-Replace        POSITION  769 SIZE 256 TYPE CHARACTER
ADD FIELD String-Return         POSITION 1025 SIZE   2 TYPE BINARY
ADD FIELD String-Start          POSITION 1027 SIZE   2 TYPE BINARY
ADD FIELD String-Length         POSITION 1029 SIZE   2 TYPE BINARY
ADD FIELD String-Times          POSITION 1031 SIZE   2 TYPE BINARY
ADD FIELD String-SearchLength   POSITION 1033 SIZE   2 TYPE BINARY
ADD FIELD String-ReplaceLength POSITION 1035 SIZE   2 TYPE BINARY
ADD FIELD String-Mode           POSITION 1037 SIZE   2 TYPE BINARY
```

To use the string-functions, add the dictionary file 'FunctionString' to the MetaMap program as a normal source-file. The type of the dictionary file is 'FUNCTION', which means that no data is read from it as a source-file, but that it is only used as an interface definition for the string-modules. The fields can be used as global variables.

Make sure that before invoking one of the modules that all input fields have a value. Use the command 'INVOKE' to call the desired module, using the function-record 'String-Parms' as a parameter.

For example: **INVOKE 'REPLSTR' (String-Parms)**

After the invoke, the output fields can be used for further processing.

17.3. INITSTR - Initialize parameter fields

```
INITSTR
       (
        String-In,
        String-Out,
        String-Search,
        String-Replace,
        String-Return,
        String-Start,
        String-Length,
        String-Times,
        String-SearchLength,
        String-ReplaceLength,
        String-Mode
       )
```

Description

Initializes all parameter fields.

Input

Not applicable

Output

String-In :	‘‘
String-Out :	‘‘
String-Search :	‘‘
String-Replace :	‘‘
String-Return :	0
String-Start :	1
String-Length :	–1
String-Times :	–1
String-SearchLength :	–1
String-ReplaceLength :	–1
String-Mode :	0

17.4. LENSTR - Determine the length of a string

```
LENSTR
      (
       String-In,
       String-Out,
       String-Search,
       String-Replace,
       String-Return,
       String-Start,
       String-Length,
       String-Times,
       String-SearchLength,
       String-ReplaceLength,
       String-Mode
      )
```

Description

Determines the length of an input string, ignoring trailing spaces.

Input

String-In	The string from which the length should be determined.
String-Length	This value should normally be set to **–1**. An improvement in performance might be achieved by setting this value to the maximum length expected. However, to get the right results, the actual length of 'String-In' must be less than this maximum expected value. (see also Note 1 under Remarks (page 137))

Output

String-Return	The returncode (see Note 2 under Remarks (page 137)).
String-Length	The length of 'String-In', ignoring trailing spaces.

Remarks

Note:

1. Essentially what happens is that the length minus trailing spaces is determined for a substring of 'String-In' starting at position 1 and with a length equal to the input value of 'String-Length'. An improvement in performance is achieved if the input value of 'String-Length' can be significantly less than 256.
2. If due to any inconsistencies in the input LENSTR is not able to determine the length, the returncode is set to **–2004** in 'String-Return'.

Examples

Focus: effect of input value of String-Length

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABC abc A'						–1				
Out					0		9				
In	'ABC abc A'						15				
Out					0		9				
In	'ABC abc A'						8				
Out					0		7*				
In	'ABC abc A'						–3				
Out					–2004		–3				

* This result is incorrect (See also Note 1 under Remarks (page 137)).

17.5. SUBSTR - Return a string from a string

```
SUBSTR
      (
       String-In,
       String-Out,
       String-Search,
       String-Replace,
```

```
    String-Return,
    String-Start,
    String-Length,
    String-Times,
    String-SearchLength
    String-ReplaceLength,
    String-Mode
  )
```

Description

Returns a substring from a given (input-)string using a start-position and length.

Input

String-In :	The string from which the substring should be taken.
String-Start :	Position in 'String-In' on which the substring starts.
String-Length :	The length of the substring to be taken. A value of **–1** indicates that the substring should be taken from 'String-In' starting at 'String-Start' and ending at the end of 'String-In', ignoring trailing spaces.
String-Mode :	The mode in which the substring should be returned. There are 3 possibilities: 0: The substring is returned as is 1: The substring is returned in uppercase format. 2: The substring is returned in lowercase format. **Note:** Other values for 'String-Mode' are treated as value 0.

Output

String-Out :	The resulting substring taken from 'String-In'.
String-Return :	The returncode (see Note 1 under Remarks (page 138)).

Remarks

Note:

1. If due to any inconsistencies in the input SUBSTR is not able to return a substring, the returncode is set to **–2001** in 'String-Return'.

Examples

Focus: effect of String-Mode and String-Length

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABCabcA'					3	3				0
Out		'Cab'			0						
In	'ABCabcA'					3	3				1
Out		'CAB'			0						
In	'ABCabcA'					3	3				2
Out		'cab'			0						
In	'ABCabcA'					3	–1				0
Out		'CabcA'			0						

17.6. SRCHSTR - Search in a string

```
SRCHSTR
        (
         String-In,
         String-Out,
         String-Search,
         String-Replace,
         String-Return,
         String-Start,
         String-Length,
         String-Times,
         String-SearchLength,
         String-ReplaceLength,
         String-Mode
        )
```

Description

Searches for occurrences of the search argument 'String-Search' in the input-string 'String-In'. The operation starts at 'String-Start' and continues over a length equal to the value of 'String-Length'. The maximum number of occurrences of 'String-Search' to look for is determined by the input value of 'String-Times'.

Input

String-In :	The string in which the search is performed.
String-Search :	The search argument.
String-Start :	Position in 'String-In' on which the search starts.
String-Length	The length over which the search is performed. A value of **–1** indicates that the search should be performed until the end of 'String-In', ignoring trailing spaces.
String-Times :	The number of occurrences of 'String-Search' to look for. A value of **–1** indicates that a search should be performed for all occurrences.

String-SearchLength : The length of the search argument. A value of –1 indicates that the search should be performed on the search argument minus trailing spaces.

String-Mode : The mode in which the search should be performed.
There are 2 possibilities:
0: The search is performed case sensitive
1: The search is performed case indifferent
Note: Other values for 'String-Mode' are treated as case sensitive (value 0).

Output

String-Return : The returncode (see Note 3 under Remarks (page 140)).

String-Start : The starting position in 'String-In' of the last occurrence of 'String-Search' that is found. 'String-Start' will contain a value of 0 if no occurrences of 'String-Search' are found.

String-Times : The number of occurrences of 'String-Search' that are actually found. This number cannot be greater than the number of occurrences that are looked for.

Remarks

Note:

1. Trailing spaces in the search argument can be included by taking the 'String-SearchLength' greater than the actual length of 'String-Search', i.e. the length minus trailing spaces.
2. If 'String-SearchLength' is less than the actual length of 'String-Search', only the first 'String-SearchLength'-part of 'String-Search' will be used to perform the search.
3. If due to any inconsistencies in the input SRCHSTR is not able to perform a search, the returncode is set to **–2003** in 'String-Return'.

Examples

- Focus: effect of String-Mode

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABCabcA'		'BC'			1	7	–1	2		0
Out					0	2		1			
In	'ABCabcA'		'BC'			1	7	–1	2		1
Out					0	5		2			

- Focus: effect of String-Times

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABAACAD'		'A'			1	7	–1	1		0
Out					0	6		4			
In	'ABAACAD'		'A'			1	7	3	1		0
Out					0	4		3			

- Focus: effect of String-SearchLength and String-Length

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABA ACA '		'A'			1	8	–1	2		0
Out					0	7		2			
In	'ABA ACA '		'A'			1	8	–1	1		0
Out					0	7		4			
In	'ABA ACA '		'A'			1	8	–1	0*		0
Out					–2003	0		0			
In	'ABA ACA '		'A'			1	–1	–1	2		0
Out					0	3		1			

* 'String-SearchLength' cannot be 0

17.7. REPLSTR - Replace substring in a string

```
REPLSTR
        (
         String-In,
         String-Out,
         String-Search,
         String-Replace,
         String-Return,
         String-Start,
         String-Length,
         String-Times,
         String-SearchLength,
         String-ReplaceLength,
         String-Mode
        )
```

Description

Replaces the search argument 'String-Search' in the input-string 'String-In' by 'String-Replace'. The operation starts at 'String-Start' and continues over a length equal to the value of 'String-Length'. The maximum number of occurrences of 'String-Search' that will be replaced is determined by the input value of 'String-Times'. The result of the operation is returned in 'String-Out'.
The length of 'String-Search' and 'String-Replace' can be different. In that case, if an occurrence of 'String-Search' is found, it will be replaced by 'String-Replace' and the remaining part of 'String-In' will be shifted to the right or to the left.
It is also possible to specify an empty string for 'String-Replace' by setting 'String-ReplaceLength' to a value of 0. This will remove 'String-Search' from 'String-In'.

Input

String-In	The string in which the search and replace is performed.
String-Search	The search argument.
String-Replace	The replace string, i.e. the string which should replace the search argument.

String-Start	Position in 'String-In' on which the search and replace starts.
String-Length	The length over which the search and replace is performed. A value of –1 indicates that the search and replace should be performed until the end of 'String-In', ignoring trailing spaces.
String-Times	The number of occurrences of 'String-Search' to replace. A value of –1 indicates that a search and replace should be performed for all occurrences.
String-SearchLength	The length of the search argument. A value of **–1** indicates that the search should be performed on the search argument minus trailing spaces.
String-ReplaceLength	The length of the replace string. A value of **–1** indicates that the search argument should be replaced by the replace string minus trailing spaces.
String-Mode	The mode in which the search should be performed. There are 2 possibilities: 0: The search is performed case sensitive 1: The search is performed case indifferent **Note:** Other values for 'String-Mode' are treated as case sensitive (value 0).

Output

String-Out	The resulting string after the search and replace is performed.
String-Return	The returncode (see also Notes 5 and 6 under Remarks (page 140)).
String-Times	The number of occurrences of 'String-Search' that are actually found and replaced. This number cannot be greater than the number of occurrences that are looked for.

Remarks

Note:

1. Trailing spaces in the search argument can be included by taking the 'String-SearchLength' greater than the actual length of 'String-Search', i.e. the length minus trailing spaces.
2. Trailing spaces in the replace string can be included by taking the 'String-ReplaceLength' greater than the actual length of 'String-Replace', i.e. the length minus trailing spaces.
3. If 'String-SearchLength' is less than the actual length of 'String-Search', only the first 'String-SearchLength'-part of 'String-Search' will be used to perform the search.
4. If 'String-ReplaceLength' is less than the actual length of 'String-Replace', only the first 'String-ReplaceLength'-part of 'String-Replace' will be used to perform the replace.
5. If due to any inconsistencies in the input REPLSTR is not able to perform a **search**, the returncode is set to **–2003** in 'String-Return'.
6. If due to any inconsistencies in the input REPLSTR is not able to perform a **replace**, the returncode is set to **–2002** in 'String-Return'.

Examples

- Focus: effect of String-Mode

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABCabcA'		'BC'	'Xy'		1	7	–1	2	2	0
Out		'AXyabcA'			0			1			
In	'ABCabcA'		'BC'	'Xy'		1	7	–1	2	2	1
Out		'AXyaXyA'			0			2			

- Focus: effect of STring-Times

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABAACAD'		'A'	'x'		1	7	–1	1	1	0
Out		'xBxxCxD'			0			4			
In	'ABAACAD'		'A'	'x'		1	7	3	1	1	0
Out		'xBxxCAD'			0			3			

- Focus: effect of String-ReplaceLength

	String-In	String-Out	Str-Srch.	Str-Repl.	Str-RC	Str-Start	Str-Lngth	Str-Tms	Str-Srch. Lngth	Str-Repl. Lngth	Str-Mode
In	'ABCabcA'		'a'	'Xy'		1	7	–1	1	3	0
Out		'ABCXy bcA'			0			1			
In	'ABCabcA'		'a'	'Xy'		1	7	–1	1	2	0
Out		'ABCXybcA'			0			1			
In	'ABCabcA'		'a'	'Xy'		1	7	–1	1	1	0
Out		'ABCXbcA'			0			1			
In	'ABCabcA'		'a'	'Xy'*		1	7	–1	1	0	0
Out		'ABCbcA'			0			1			

Any other value for 'String-Replace' would have the same result since 'String-ReplaceLength' is set to 0.

CHAPTER 18

Calling the Generator in Batch

The Generator is called in batch by running the *MSBGEN.EXE* program from the DOS prompt. This program is available in the MetaSuite installation folder.

It allows:

- creating or maintaining the Generator Dictionary,
- regenerating MXL files.

18.1. Preparation

Before running the MSBGEN.EXE program, copy the command file, e.g. a MIL or MXL file, to the *TMP* temporary folder under <*InstallationFolder*>/*GEN**. Then rename the file to *PPTOUT.TMP*. *PPTOUT.TMP* is the normal output of *MetaMap* and will be used as input for *MSBGEN.EXE*.

18.2. Using MSBGEN.EXE

After the User has called MSBGEN.EXE, output information of the generating process can be found in PPTIN.TMP. PPTIN.TMP is input for MetaMap and output of MSBGEN.EXE.

Open an MS-DOS session and browse to the MetaSuite installation folder.

Run the program using the following command format:

```
MSBGEN generation-type information-mode directory-information
```

The following table provides an overview of the available generation types.

Type	Description
MIL	MetaSuite installation script, written in MIL language.
MXL	MetaSuite export script, written in MetaSuite Definition Language and MetaSuite Specification Language.
MDL	MetaSuite Definition Language, just for testing purposes.
MSL	MetaSuite Specification Language, just for testing purposes.

The following table provides an overview of the available information modes.

Mode	Description
NOTRD	Standard mode, no trace mode, no debug info.
TRACE	Trace mode.
DEBUG	Debug information in order to see which tables the Generator generates.
TRDBG	Combination of Trace and Debug modes.

The following table provides an overview of the possible directories.

Directory	Description
Temporary	Temporary directory for putting temporary files. The file *mscopy.mil* will be put here also, when the MIL command "Copy Table" has been used.
MGL directory	This directory will contain the MGL files, if you generate an MXL or MSL file.
MRL directory	This directory will contain the MRL files, if you generate an MXL or MSL file.
DCT directory	This is the directory where MetaSuite can find or has to store the Generator Dictionary File (msdict.dct).
DDL directory	This is the directory where MetaSuite can find the source files to feed the Generator Dictionary File (*msini.ddl*,*msmgl.ddl*and*msmrl.ddl*). This directory has to specified if the MIL command NEW is used.
Installation directory	Folder name (short name) where MetaSuite is installed. This directory must be specified if the MIL command NEW is used.

18.3. Example

1. Copy an MXL file to <InstallationFolder>\GEN*\TMP.

2. Rename the file to pptout.tmp.

3. Enter the following command at the DOS prompt:

```
MSBGEN MXL DEBUG '<Doc>\GEN*\TMP' '<Doc>\GEN*\MGL' '<Doc>\GEN*\MRL'
'<Doc>\GEN*\DCT' '<InstallationFolder>\GEN*\DLL'
'<InstallationFolder>’
```

For example:

```
SET MSINS=D:\Program Files\Ikan Solutions\MetaSuite 813
cd /D "%HOMEDRIVE%%HOMEPATH%\Documents\Metasuite"
copy MXL\%1.mxl .\GENIBM_Windows\TMP\pptout.tmp
"%MSINS%\msbgen.exe" MXL NOTRD ^
  ".\GENIBM_Windows\TMP" ^
  ".\GENIBM_Windows\MGL" ^
  ".\GENIBM_Windows\MRL" ^
```

```
  ".\GENIBM_Windows\DCT" ^
  "%MSINS%\GENIBM_Windows\DDL" ^
  "%MSINS%"
echo %errorlevel%
```

4. Browse to the *<InstallationFolder>\GEN*\TMP* folder.
   Verify the file *pptin.tmp* with a text editor (e.g., Notepad).

5. Browse to the *<InstallationFolder>\GEN*\MGL* folder.
   Verify the file *EX0.MGL* to see the generated COBOL source.

6. Browse to the *<InstallationFolder>\GEN*\MRL* folder.
   Verify the file *EX0.MRL* to see the generated run-time script.

CHAPTER 19

Generator Message Reference

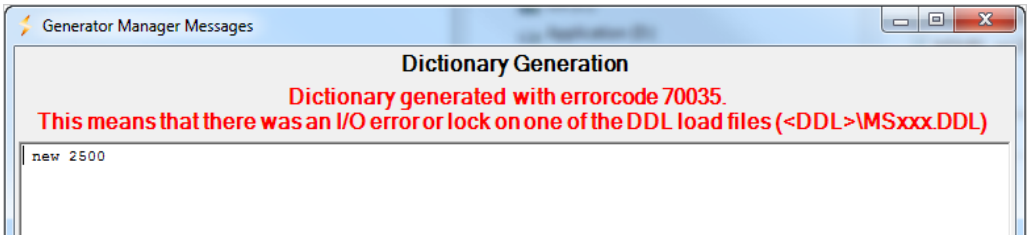
This section describes the following:

- Generator Return Codes (page 147)
- Generator Syntax Error Messages (page 150)
- Dictionary Error Messages (page 151)
- List of Generator Messages (page 151)

19.1. Generator Return Codes

Generator return codes identify potential problems with the Generator system files which occur at generation time.

Generator Return Codes are displayed in a dialog box with the following format:


Generator Manager Messages
Dictionary Generation
Dictionary generated with errorcode 70035.
This means that there was an I/O error or lock on one of the DDL load files (<DDL>\MSxxx.DDL)
new 2500

Call MetaSuite technical support and keep a copy of the file causing the problem at hand for further investigation.

Generator Return Codes are always composed of 5 digits. The leading digit refers to the file(s) concerned, as indicated in the table below.

First digit	File(s) concerned
1	Generator output list: **TEMPDIR\PPTIN.TMP**
2	Dictionary upload file at installation time: **<InstallationFolder>\...\DCT\MSDICT.DCT**
3	Dictionary upload file at generation time: **<InstallationFolder>\...\DCT\MSDICT.DCT**
4	Command file (MXL or MIL file type): **TEMPDIR\PPTOUT.TMP**
5	One of the generated files, MGL or MRL: **<InstallationFolder>\...\MGL\mxlname.MGL** **<InstallationFolder>\...\MRL\mxlname.MRL**
6	Backup file: **TEMPDIR\mscopy.mil**
7	One of the DDL load files, INI, MGL or MRL: **<InstallationFolder>\...\DDL\MSINI.DDL** **<InstallationFolder>\...\DDL\MSMGL.DDL** **<InstallationFolder>\...\DDL\MSMRL.DDL**
8	Dump folder at generation time. This folder is used for putting the requested dump file when the MTL command ">DUMP" is executed. Default: **TEMPDIR**

Note: In the references above, **TEMPDIR** points to the directory defined in the TEMP system/user variable.

The four following digits match one of the following codes:

Four-digit code	Description
0002	Return code only applicable for indexed files. Meaning: • For a READ statement, the value for the current key is equal to the value of that same key in the next record within the current key of reference. • For a WRITE or REWRITE statement, the record just written created a duplicate key for at least one alternate record key for which are allowed.
0004	The length of the record being processed does not conform to the fixed file attributes for that file.
0005	The referenced optional file is not present at the time the OPEN statement is executed.
0007	Sequential files only. For an OPEN or CLOSE statement with the REEL/UNIT phrase of the referenced file is a non-reel/unit medium.
0010	No next logical record exists. You have reached the end of the file.
0014	Relative files only. The number of significant digits in the relative record number is larger than the size of the relative key data item described for that file.

Four-digit code	Description
0021	For sequentially accessed files only: indicates a sequence error. The ascending key requirements of successive record key values has been violated, or, the prime record key value has been changed by a COBOL program between successful execution of a READ statement and execution of the next REWRITE statement for that file.
0022	For indexed and relative files only, indicates a duplicate key condition. Attempt has been made to store a record that would create a duplicate key in the indexed or relative file OR a duplicate alternate record key that does not allow duplicates.
0023	Indicates no record found. An attempt has been made to access a record, identified by a key, and that record does not exist in the file. Alternatively START or READ operations have been tried on an optional input file that is not present.
0024	Relative and indexed files only: Indicates a boundary violation arising from one of the following conditions: • An attempt is made to write beyond the externally defined boundaries of a file. • A sequential WRITE operation has been tried on a relative file, but the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file.
0030	The I/O statement was unsuccessfully executed as the result of a boundary violation for a sequential file or as the result of an I/O error, such as a data check parity error, or a transmission error.
0034	The I/O operation failed because of a boundary violation. This condition indicates that an attempt has been made to write beyond the externally defined boundaries of a sequential file.
0035	An OPEN operation with the I-O, INPUT, or EXTEND phrases has been tried on a non-OPTIONAL file that is not present.
0037	An OPEN operation has been tried on a file which does not support the open mode specified in the OPEN statement.
0038	An OPEN operation has been tried on a file previously closed with a lock.
0039	A conflict has been detected between the fixed file attributes and the attributes specified for that file in the program.
0041	An OPEN operation has been tried on file already opened.
0042	A CLOSE operation has been tried on file already closed.
0043	Files in sequential access mode: the last I/O statement executed for the file, before the execution of a DELETE or REWRITE statement, was not a READ statement.
0044	A boundary violation exists.
0046	A sequential READ operation has been tried on a file open in the INPUT or I-O mode but no valid next record has been established.
0047	A READ or START operation has been tried on a file not opened INPUT or I-O.
0048	A WRITE operation has been tried on a file not opened in the OUTPUT, I-O, or EXTEND mode, or, on a file open I-O in the sequential access mode.
0049	A DELETE or REWRITE operation has been tried on a file that is not opened I-O.

If the Return Code = -1, there has been a subsystem error at generate time.

19.2. Generator Syntax Error Messages

This section lists the messages signalling problems with Generator commands at generation time. If the MetaSuite Generator is launched from within MetaMap, the messages will appear in the MetaSuite Generator part of the message window within MetaMap.

Generator Syntax Errors occur very rarely if the MetaSuite Model has been developed using MetaMap.

The actual error message can be preceded by a message in the following format:

SYNTAX ERROR m OCCURED NEAR : xxx

Where:

- m = sequential error number
- xxx = string where the error occurred

Example:

SYNTAX ERROR 1 OCCURED NEAR : XIZE

The error message itself has the following format:

PgmNnnL m Message_Text

where:

- Pgm = reference to the module PPTPgm.DLL that handles the command.

  Note: Those first 3 characters are indicators that can be uselful for technical support.
- Nnn : General Message number
- L : The error level
- m : the error count for the current MXL statement
- Message_Text : error description

Example:

FES109E 1 KEYWORD EXPECTED

The following error levels exist:

Error level	Description
I (Informational)	An informational message provides useful information.
W (Warning)	A warning indicates an inconsistency in the command. The system will take some type of default corrective action (usually indicated by the message), and the command will be processed. You should check all warning messages to ensure that the command was processed in the way intended.
E (Serious error)	A serious error indicates an error that makes it impossible for the system to process the command. When the first serious error is encountered, program generation is halted, but syntax checking continues for the rest of your program code. You will need to re-code and re-enter the command in error.
F (Fatal error)	A fatal error indicates that either the system's security facility did not clear your user identification, or that there is a serious problem with your dictionary. In either case, your run will be terminated immediately, with no further processing of commands. In the former case, either provide the proper identification or see your System Administrator to establish a valid password and user ID.

19.3. Dictionary Error Messages

Dictionary Error Messages refer to errors caused by a faulty or inconsistent dictionary. These errors are not numbered because the numbered messages themselves are part of the dictionary. If the dictionary is faulty, the numbered messages are not accessible.

The following table gives an overview of the Dictionary Error Messages:

Message(s)	Meaning
F DICTIONARY I/O ERROR ON KEY 1 1 0 F FAILED I/O FUNCTION READ F RUN ABORTED	There is an inconsistency in your dictionary. Ask your MetaSuite administrator to rebuild the dictionary.
F DICTIONARY UPDATE FLAG FOUND ALREADY SET WHEN DICTIONARY WAS OPENED. EITHER CURRENTLY IN USE, OR A PREVIOUS UPDATE JOB HAS BEEN INTERRUPTED.	The dictionary has been deleted or corrupted. Ask your System Administrator to restore the dictionary.
F DICTIONARY INCOMPATIBLE WITH : Generator-version F PLEASE UPGRADE DICTIONARY TO : Dictionary-version	The dictionary is not compatible with the generator, because it is outdated. Upgrade the dictionary to another build number or to another version number. The minimum version and build number is indicated in this message.
F DICTIONARY INVALID	The dictionary is not compatible with the generator, but the cause cannot be determined. Create a new dictionary.

19.4. List of Generator Messages

This section contains a complete list of Generator Messages. For easy reference, they have been grouped in sections each containing 50 messages.

- Messages 001 to 050 (page 152)
- Messages 051 to 100 (page 154)
- Messages 101 to 150 (page 157)
- Messages 151 to 200 (page 160)
- Messages 201 to 250 (page 163)
- Messages 251 to 300 (page 165)
- Messages 301 to 350 (page 167)
- Messages 351 to 400 (page 169)
- Messages 401 to 450 (page 171)
- Messages 451 to 500 (page 174)
- Messages 501 to 550 (page 176)
- Messages 551 to 600 (page 178)
- Messages 601 to 650 (page 181)
- Messages 651 to 700 (page 184)
- Messages 701 to 711 (page 190)

Messages 001 to 050

Number	Message	Explanation
001	Date not in Y2K format	This warning message is returned for each non-Y2K date field used within a program.
002	Name begins with an invalid character	The name of the object is not according to the naming standards: the first character is wrong.
003	Name contains an invalid character	The name of the object is not according to the naming standards: one of its characters is wrong.
004	Name length too long	
005	Quotes must enclose literal	
006	Name cannot end in a hyphen	A name of an object is ending in a hyphen, which is forbidden.
007	Numeric contains an invalid character	
008	Too many decimal points	A decimal value containing more that one decimal point (.) is being provided.
009	Operator is not followed by a space	If an operator is not followed by a space, this message is returned. You should always add a space after the operator (which will be = + - * / **).
010	Name already defined in Dictionary	The object to be added has a name that already exists in the technical dictionary. Each object within the technical dictionary should have a unique name. You can either rename the object yourself, or you can put in the beginning of the maintenance file the command 'DUPLICATES'. MetaSuite Generator will then generate a new name for the duplicate object name by adding a suffix.
011	Dictionary I/O error	If an object is not defined in the dictionary, this message will be returned. This message will often by accompanied by message number 35. Please check the name of the object you entered, or create a list of all existing objects in the technical dictionary to know which objects are defined in the technical dictionary.
012	#1 Not defined in Dictionary	The generator dictionary msdict.dct cannot be accessed for an unknown reason. Check the access rights (write access), the file availability, the network connection etc. If no external reason is found, it is possible that the dictionary is full or corrupt. In that case: Please remove the file and build a new dictionary.
013	Invalid command sequence	A MetaSuite program is structured according to a certain sequence of 4 parts. If the sequence of commands within the MSL is not respected, this message is returned.
014	End of Command or REMARKS expected	If there are too many arguments on a command, this message is returned.

Number	Message	Explanation
015	Duplicate keyword	If a keyword in a statement is used twice (or more), this message is returned. You should remove one of the occurrences of the keyword (with its characteristics).
016	File name expected	The ADD FILE statement has no name specified, or has an invalid name specified.
017	Name expected	This general message is returned if no name or an invalid name is specified on an object. You must build up maintenance commands with MetaStore Manager and programming commands with MetaMap.
018	Incomplete command	This general message is returned if some mandatory keywords are not provided in the command. You must build up maintenance commands with MetaStore Manager and programming commands with MetaMap.
019	Invalid use of keyword	
020	Numeric argument expected	If a keyword expects a numeric value, this message will be returned.
021	Field name expected	This message is accompanied with the message *SYNTAX ERROR n OCCURED NEAR : xxx*. It expects *xxx* to be a field name, which is not the case.
022	File key not allowed	
023	File type invalid (or missing)	If the TYPE of a FILE is not one of the known file types, or if the file type is missing, this message is returned.
024	HALT not allowed on structures or FUNCTION files	The HALT statement cannot be applied on a file that is not opened nor read. Halting a file means that the process of reading the file is halted. Hence: if the file is not read, no HALT can be done.
025	Maximum record size limit reached	
026	Position required relative to	
027	Mode invalid or not supported	If the MODE of a file is not EBCDIC or ASCII, or if the specified MODE is invalid for the chosen MetaSuite Generator, this message is returned.
028	VALUE must be larger than zero	Certain characteristics can not have the value 0 assigned. Please change the value to a proper value.
029	Must be an integer value	
030	List option invalid	
031	Record key value conflict with type	
032	Table name must be 6 characters long	A table containing template code for the technical dictionary (installation commands) should have a name of exactly 6 characters.
033	Left parenthesis expected	
034	Conflicts with prior entry	

Number	Message	Explanation
035	No valid definitions requested	If there are no resulting objects for a list of objects, this message is returned.
036	Too many definitions requested	
037	Numeric contains too many digits	
038	Fraction missing	
039	Literal improperly delimited	If an alphanumeric constant is not properly enclosed between singled quotes, this message is returned.
040	Date mask usage conflicts with	
041	This is not allowed after SHORT	Syntax error in the FIELDS command within a report.
042	DBNAME value too long	The value of DBNAME is exceeded. The number of characters allowed depends on the database type.
043	HALT <target> automatically replaced by HALT ALL	A target file that is halted does not make the source files halt. Skipped source file records will be treated as "excluded" records. So, in case of HALT <targetfile x> all source files loop until they reach the end (EOF condition). If the only source files are function files and special write only files (which are not read), parameter files and arrays (which are read & closed before main processing) this EOF condition never becomes true, and the program will loop endlessly. In order to avoid such behavior, HALT <targetfile x> is changed into HALT ALL.
044	Keyword or name expected	
045	Comma expected	
046	EXCLUDE not allowed	
047	Literal or name expected	If the literal or name does not contain a correct value, this message is returned.
048	Default key field size exceeded by	Something is wrong with the length of the key field.
049	Remaining entries ignored	If a previous error has occurred, the following keywords on the command are ignored. You first have to correct the previous error before you can continue.
050	---	

Messages 051 to 100

Number	Message	Explanation
051	Comma or right parenthesis expected	
052	Incorrect License Key	This message appears if you did not add a license key, or if you added an invalid license key. The current license may also be expired.

Number	Message	Explanation
053	Decimal places conflict with	Only numeric data types can have a DECIMAL indication. If a DECIMAL is used on a field definition where it should not be used, this message is returned.
054	Right parenthesis expected	If a right parenthesis is expected, but not present, this message is returned. A right parenthesis is used to end a logical set of MetaSuite commands, such as a DETAIL or TOTAL line on a target file. This error is usually related to a previous error, which makes the MetaSuite Generator ignore the right parenthesis which is present. You must first correct the previous error.
055	Only one character allowed	
056	#1 not allowed within FOR-block. END-FOR required.	This statement is not allowed until the FOR-block is closed.
057	NEXT is an obsolete statement	It is advised to structure this procedure instead of using an older expression such as NEXT.
058	Transformation object:	This message is used for outputting the transformation numbers if generation is performed in TRACE mode.
059	Alias Table is full	Record names, suffixed with the "#"-sign and a field number (generated by MetaMap) are internally replaced by an alias name. The number of alias names is limited to 500.
060	This feature is not supported anymore.	
061	File containing this field is not in use	
062	Assignment symbol expected	An assignment symbol is considered to be = (the equal sign) followed by a space. When only the = is present and the value to be assigned is concatenated to the =, the = will not be considered as an assignment symbol.
063	Record size invalid	The record size is zero, negative or higher than the file size.
064	CODE clause incompatible with #1	When a number is given the attribute "CODE", other specifications like "NATIVE", "TIMESTAMP" etcetera will overrule this "CODE" specification.
065	HALT statement needed in order to avoid endless loop	If the only source files are function files and special write only files (which are not read), parameter files and arrays (which are read & closed before main processing) the EOF condition never becomes true, and the program will loop endlessly. In order to avoid this, a (conditional) HALT statement should be programmed somewhere, preferably in a TARGET procedure.
066	VALUE is not within the range of field limits	On a field where limits are defined, values should be within these limits. When the value is not within the limits, this message appears.
067	Inital values cannot be used with	
068	Range invalid	
069	Maximum of 16 records exceeded	

Number	Message	Explanation
070	Too many values	
071	Too many SORT or GROUP fields	
072	Heading literal exceeds 29 characters	
073	Value longer than field size	A value is assigned to a field that is too big corresponding to the field size.
074	#1-structure automatically closed	When meeting END-FOR, the IF- or CASE-instructions that have not been closed properly, will be closed automatically.
075	---	
076	'Based-on' field must be defined in	
077	Name not unique within the Dictionary	The object that is to be added has a name that already exists in the technical dictionary. Each object within the technical dictionary should have a unique name. Since the 'DUPLICATES' command is used in the beginning of the maintenance statements, this is given as a warning since the name will be renamed automatically. It is always associated with the following message which says which rename has been done.
078	Name has been changed to	New Object_name Since the 'DUPLICATES' command is used in the beginning of the maintenance statements, all non-unique names are renamed by MetaSuite Generator. This message will give you the new name for the object.
079	Invalid use of system keyword	
080	Dictionary file #1: #2	
081	Invalid use of COBOL reserved word	
082	Limits cannot be used with	
083	Equate values cannot be used with	
084	Index must refer to a field	
085	Field size invalid	Certain data types expect only a limited field size. If there is a conflict on the field size, this message is returned.
086	Already defined within	
087	The block size must match	
088	---	
089	Date mask invalid	
090	Dictionary command invalid	If the first command of a maintenance statement is not correct, this message is returned.

Number	Message	Explanation
091	Definition type invalid	The object type on which the primary action of the maintenance command is to be performed is invalid. A maintenance command always consists of Action_To_Be_Performed Object_type Object_name Object_Characteristics With Object_type either [FILE | RECORD | FIELD | INDEX | LINK]
092	Command invalid	General message given when a command within a procedure (source file procedure / target file procedure / report procedure) is incorrect. Further information on what is incorrect is given in previously given messages.
093	Size of the spanning item is not large enough	Size for ObjectName is not big enough compared with the fields defined underneath. The ObjectName can be the name of a file, a record or a field with subfields. Please check the sizes and correct them.
094	Field type invalid	The data type of a field is incorrect.
095	Bit position exceeds character size	
096	Decimal places exceed field size	If there is a SIZE problem in your field definition, the DECIMAL indication is no longer correctly taken and this message will be returned.
097	Module name too long	
098	Must be an Infile Initial Procedure	
099	Equate longer than 57 characters	
100	Record name expected	

Messages 101 to 150

Number	Message	Explanation
101	STANDARD or OMITTED required	
102	User labels not permitted	
103	Duplicate Sort or Group Key	
104	Controlling file does not precede controlled file	

Number	Message	Explanation
105	File must have a Key	Certain file types or MSL commands need to have a key defined on the file (e.g. Indexed Files, External Arrays, Controlled files). If no key is defined on the file, this message will be returned. **Example 1:** ADD FILE XXX TYPE INDEX FIX 10 BLOCK 15 **Example 2:** GET ARR-employee-master KEY = EMPLOYEE-NUMBER#00053 **Example 3:** SOURCEFILE DEPARTMENT-ARRAY CONTROLLED BY employee-master \Syntax Error 1 Occurred Near: employee-master KEY DEPARTMENT#06502 \FIG105E 1 File Must Have Key This is an example with a sequential file that is at the same time a CONTROLLED file. The *Automatic* check box was cleared. No key value search can be done because it is a sequential file that was not read in as an external array. On the other hand, a *controlled by* structure needs a key search. This is a conflicting situation, resulting in this error message.
106	Record size exceeds block size	The block size on a file should be bigger than the maximum record size. If the block size is smaller than the maximum record size, this message is returned.
107	Maximum record size required	The ADD FILE does not have a maximum record size specified through the FIXED, VARIABLE or UNDEFINED keyword
108	SPANNED not allowed	This error will occur on the ADD FILE statement when the SPANNED option is not allowed, but is specified.
109	Keyword expected	There is a mandatory keyword missing on the statement.
110	Numeric or literal expected	A constant value is given in the statement which is not according to the following standards: a non-quoted value should be numeric, an alphanumeric value should be put between single quotes.
111	Literal expected	
112	Must be less than 100	
113	Numeric or field name expected	
114	Record or field name expected	The ADD FIELD statement should specify a known owner field or a known owner record. The owner object specified on the ADD FIELD statement is not a known record or field.
115	must be defined within #1	
116	SORT field larger than allowed	The maximum size of the SORT field is 256 characters.
117	Must not exceed page size	
118	Edit character invalid	If an edit mask is given and it contains an invalid character, this message is returned. This could happen when a given date format contains an invalid character.

Number	Message	Explanation
119	Placement of edit character invalid	If the edit mask of a field is incorrect, this message is returned. Please verify the field's edit mask.
120	Sign placement invalid	
121	Trailing sign must be followed by a quote	If the edit mask of a numeric field contains the trailing sign symbol not at the beginning of the edit mask, the sign should be followed by the ending quote for the edit mask. If this is not the case, this message is returned.
122	Rightmost characters truncated	If the edit mask given on an alphanumeric work field is smaller than the size given for the work field, this message is returned. It is a warning message telling you that the size of the edit mask (which is smaller than the work field size) will be respected.
123	Most significant digits truncated	If the size of a field is not big enough for the values assigned to the field (e.g. lower limit, higher limit, initial value), this message is returned. If the edit mask given on a numeric work field is smaller than the size given for the work field, this message is returned. It is a warning message saying you that the size of the edit mask (which is smaller than the work field size) will be respected.
124	Least significant digits truncated	If the edit mask given on a numeric work field with decimals does not contain a decimal part, this message is returned. It is a warning message saying you that the edit mask (with no precision given) will be respected.
125	Space not allowed	
126	File name required	The record to be added can not be added to a file, since no file is active. This can happen when the previous 'ADD FILE' statement has been unsuccessful, and no owning FILE has been specified for the record.
127	Record name required	If there is no current record, all fields should have an owner record specified. If this is not done, this message is returned.
128	Field position required	There is no POSITION specified for the field. When the record for the field is known this does not give any problems. Example: ADD FIELD C TYPE MIXED
129	Value conflicts with field type	A value is given in the statement, but it is not according to the expected data type. This could happen on the initial value of an alphanumeric field, which is set to a numeric value.
130	Value conflicts with sign	
131	Size required	There is no SIZE specified on the definition of the field.
132	File command required	
133	TARGETFILE command required	
134	BEGIN command missing	If MetaSuite Generator expects a command to start with the keyword BEGIN (for all procedure definitions), but the command is not present, this message is returned.

Number	Message	Explanation
135	Procedural command missing	
136	Internal name limit exceeded by #1	
137	Cannot define dependent field	
138	List incomplete	If a command expects to have a list of objects, but the list of objects is not complete according to the syntax, this message is returned.
139	List entry expected	
140	Maximum of 32 values exceeded	If more than 32 arguments are given for the INVOKE or DEBUG command, this message is returned.
141	Field is undefined	The field xxx is not defined in the MetaSuite program as either a source field, a global field or a target field.
142	Field must be subscripted	If a source field belongs to a source record that is occurring, or if a global field or a target field is an occurring field, and there is no subscript given when the field is referenced, this message is returned.
143	Subscript exceeds occurrences	This message is returned for fields that are to be subscripted.
144	Additional subscript(s) required for	This message is returned for fields that are to be subscripted.
145	Too many subscripts for	This message is returned for fields that are to be subscripted.
146	Subscript may nog contain an expression	This message is returned for fields that are to be subscripted.
147	Subscript must be numeric	This message is returned for fields that are to be subscripted.
148	Subscript may not be subscripted	This message is returned for fields that are to be subscripted.
149	Subscript must be larger than zero	This message is returned for fields that are to be subscripted, but for which the subscript value is zero or undefined.
150	Subscript expected	This message is returned for fields that are to be subscripted.

Messages 151 to 200

Number	Message	Explanation
151	Incomplete subscript	If a global field or a target field is an occurring field, and no subscript is given when the field is referenced, this message is returned.
152	First operand not found	

Number	Message	Explanation
153	Comma ignored	
154	Expected entry not found	If a command is not correctly build up (there is an entry missing), this message is returned. Check your command to detect what entry is missing.
155	Not a valid system field name	This message is accompanied by *SYNTAX ERROR n OCCURED NEAR: xxx* where *xxx* indicates what token is expected to be a system field, but which is an invalid system field name.
156	System field name expected	When in a command a system field is expected, but the end of the command is reached at this place this message appears.
157	Subscript must be an integer	
158	System field may not be subscripted	
159	Subscript or expression ignored	If a field in a MetaSuite program has incorrectly defined a subscript, this message will be returned.
160	List may not contain an expression	
161	Function has no parameters	
162	Date field expected	
163	Command not in Target File Procedure	
164	Invalid date	
165	Command(s) following EXIT ignored	All commands following the EXITcommand will not be executed.
166	Command(s) following EXCLUDE ignored	All commands following the EXCLUDE command, will be ignored.
167	Name must be unique	
168	Option conflics with prior option:	
169	---	
170	Edit line is undefined	
171	Workfield can sort only one file	
172	Field not defined on Input File	
173	Line number expected	
174	Must not be less than zero	
175	Line number must be in sequence	

Number	Message	Explanation
176	Edit line information incomplete	If the end of a logical set of MetaSuite commands, such as a DETAIL or TOTAL line on a target file, is not present, this message is returned. This error is usually related to a previous error, which makes the MetaSuite Generator ignore the end command of the logical set. You must first correct the previous error.
177	Option not allowed	
178	Edit line may not contain an expression	
179	Number of lines to be skipped expected	
180	Postion number expected	
181	Postion may not be specified	
182	TARGETFILE number expected	
183	Source file is not being used	If a Source File is used which is either not defined as Source File, or which is not correctly defined as Source File, this message is returned.
184	Target file is not being used	
185	Postion expected	
186	Must be preceded by an edit value	
187	Keyword AND missing	If a concatenation is expected (done by the keyword 'AND') this message is returned. Often this message is returned on a statement on which a previous error has occurred. First correct the previous error.
188	Conflict with 'No auto' option	
189	Conflict with 'No auto' option (no group/accum fields)	
190	Maximum of 12 columns exceeded	
191	Maximum number of columns exceeded	
192	TARGETFILE number must be higher than previous	
193	TARGETFILE number higher than 99	If more than 99 Target Files or more than 99 Reports are added to a program, this message will be returned.
194	File already in use	
195	Unknown function #1	
196	Relational operator expected	When a conditional expression is built up, but there is no relational operator where it needs to be, this message is returned.
197	Keyword TO required	

Number	Message	Explanation
198	Unbalanced parentheses	
199	Operands must be of the same class	If a comparison is done, and the operands in the comparison are not compatible, this message is returned. Please check the data types of the operands and correct them if necessary.
200	Rest of expression ignored	If a previous error has occurred on a statement, this message is returned. Please first correct the previous error, to check the full expression.

Messages 201 to 250

Number	Message	Explanation
201	---	
202	Edit line overflow	
203	Prefix must be 4 characters long	
204	No fields defined for #1	At least one field must be defined in this file or record.
205	Double/triple subscript not allowed	
206	Number of match keys invalid	If the compatibility rule on matching Source Files is not correct (the matching Source Files do not have the same number of match fields defined), this message is returned.
207	---	
208	Match key of #1 must be a workfield or	
209	No records defined for file	
210	Keyfield is not defined	
211	Not accessible from this procedure	
212	Default title longer than page width	
213	Counter field must be a workfield	
214	IF may not be subordinate to CASE	
215	Must be preceded by an assignment value	
216	Inconsistent data type	If you want to assign fields of incompatible data types to one another, this message is returned. The following assignment would give this message, since you can not assign an alphanumeric value to a numeric field.

Number	Message	Explanation
217	---	
218	Too many decimal places	
219	Concatenation not allowed	
220	Field must be numeric	
221	Sample command invalid	
222	Operand expected	
223	Arithmetic operand expected	Each operator always needs to be followed by a space. If this space is not present, or if there is an operator missing in an arithmetic expression, this message is returned.
224	Expression may not be concatenated	
225	Expression incomplete	If an arithmetic expression contains an error, this message will be returned. It could simply be because the arithmetic operator is not recognized if it is not followed by a space.
226	Must be preceded by an operator	
227	Must be preceded by an operand	
228	Procedure not defined	
229	Name already in use	
230	Record #1 is not defined	
231	Edit value invalid	
232	Assignment value expected	If a command contains the assignment symbol (=) and no value follows, this message is returned.
233	Transformation label:	If a program is generated in trace mode, this message will be returned for each calculation within the program.
234	---	
235	Edit value expected	
236	Procedure name expected	
237	Must precede Target File Procedures	
238	BEGIN operand invalid	
239	Procedure type invalid	
240	Duplicate procedure	If two procedures have been defined with the same execution time on the same Source File or Target File, this message is returned. You can only have one procedure with a certain execution time on the same Source File or Target File.
241	Erroneous prior reference exists	

Number	Message	Explanation
242	Must be previously referenced	If a procedure is defined in the MSL, and this procedure is not used in one of the other procedures (recognition is done on name basis), this message is returned.
243	Procedure is out of sequence	
244	Must be in the range of 1 to 9	
245	SKIP number exceeds page size	
246	Edit values may not overlap	
247	---	
248	---	
249	---	
250	File already controls another file	

Messages 251 to 300

Number	Message	Explanation
251	---	
252	---	
253	---	
254	Field must be a Workfield	
255	---	
256	---	
257	---	
258	---	
259	---	
260	Subscripted reference not allowed	
261	Must be a totalled field or a workfield	
262	Value must be moved to a workfield	
263	Title + Heading + Endpage exceeds #1 lines	
264	Field name sequence invalid	
265	Edit line type not allowed	
266	Field cannot be totalled	
267	Skip not allowed on title line 0	

Number	Message	Explanation
268	Function not allowed	
269	Function ignored	
270	Function incomplete	
271	Arithmetic operator invalid	
272	Date field not allowed	
273	Operand list not allowed	
274	Literal size exceeds 58 characters	
275	Forked path invalid in	
276	Not a valid field name	
277	Reserved word not allowed	
278	---	
279	Literal of length zero	
280	Invalid COBOL identifier name	
281	Name has been truncated to	
282	COBOL picture clause is invalid	
283	`Based-on´ field not defined	
284	EXCLUDE file name invalid for #1	
285	Record not in lowest level of path	
286	File must have a Path	
287	Value invaled - changed to spaces	
288	Only allowed in File Initial or File Input Procedure	The mentioned command can only be used in a Source File initial procedure or a Source File input procedure. This message is returned if the procedure in which the command occurs is not a a Source File initial procedure or a Source File input procedure.
289	Not added	
290	Not deleted	
291	Invalid option for database file type	
292	Confidence level lower than 80%	

Number	Message	Explanation
293	'File Path' or 'Get Recordname' Required	If a command is issued on a Source File which needs to know a Source Record name, this message is returned if the Source File is not defined with a path. You either issue the command with the Source Record name directly, or you add a path to the Source File, sothat the Source Record on which the command must be issued is defined as entry record in this path.
294	---	
295	Key clause required for this file	
296	Optinal feature (password expected)	
297	Option ignored for workfield	
298	Replacing `/´ with `B´	
299	PATH or CONTROLLED required	
300	Table name expected	

Messages 301 to 350

Number	Message	Explanation
301	Record #1 not defined in File Path	This error is issued when the record referred to by this sentence is not found in the path declaration of the current file. The file must be buffered and the record mentioned here must be part of the file buffer.
302	---	
303	---	
304	Not allowed with SORT or within GROUP	
305	---	
306	May not exceed 30 characters	
307	#1 May not contain quotes	
308	Record exists, cannot change mode	
309	Link requires FROM and TO records	
310	Links FROM and TO Record are the same	
311	Link name expected	
312	FROM clause must precede TO clause	

Number	Message	Explanation
313	Name of storage area required for #1	
314	Index name expected	
315	'Based-on' field name required	
316	---	
317	---	
318	No data referenced for #1. This might lead to compile errors.	This error is issued when no data of a certain file has been referred to. The file is not used.
319	---	
320	File must be an Input File	
321	---	
322	---	
323	---	
324	---	
325	Size must be 2, 4 or 8	
326	---	
327	OCCURS not allowed for #1	
328	Link or index name expected	
329	VIA not allowed at level 1	
330	Sort, Pre-pass or Extract required	
331	---	
332	Not a key for this file	
333	File security prevents the usage of	
334	Does not provide an index on	
335	Does not link to prior path record	When a path statement for a Source File uses a link between 2 records, which is not defined between those 2 records, this message will appear. Please verify how the link is defined in MDL (between which records is the link defined) and check how the link is used in MSL.
336	Does not appear prior in path	
337	Size must be in the range of 1 to 10	
338	No links found which provide path to #1	
339	Ambiguous path (link name needed) for	

Number	Message	Explanation
340	Invalid VIA record (Occurs) for #1	
341	Too many (more than 15) subpaths	
342	Not allowed for TARGETFILE command	
343	Only 1 value allowed in Target File	
344	Maximum number of target fields exceeded	
345	Sequence of records must be reversed	
346	No Detail/Total lines defined	When there are no detail or total lines defined for a Target File, or when the detail or total lines for the Target File are not found due to a previous syntax error in the Target File definition, this message appears. Please check first whether there is a previous syntax error which causes the problem. If not, please return to MetaMap and add a TargetRecord of type Detail or Total.
347	Invalid for a non-numeric field	
348	`OF Recordname´ required	
349	Path specifies associated record twice	
350	Keys are not allowed for this file type	If the KEY is used for a Source File which is not correctly defined, this message is returned.

Messages 351 to 400

Number	Message	Explanation
351	File cannot be controlled	
352	---	
353	Field must be within START File	
354	Ambiguous reference	
355	Prefix required	
356	---	
357	Path Index name required for START	When a START command with a KEY is used on a Source File (used for non relational source files), the entry record within the path must be accessed by an index (to enable the direct access of the START command). When the entry record is not accessed by an index this message is generated.
358	---	

Number	Message	Explanation
359	End of Command or OFF expected	
360	Literal or OFF expected	
361	Prefix can be max. 12 characters long	
362	Consecutive hyphens not allowed	
363	---	
364	---	
365	No DBNAME defined for	
366	Format list overflow in file	The usage of the format list code is reserved for internal use. Please contact your MetaSuite administrator or IKAN Solutions N.V.
367	No storage key defined for	
368	---	
369	---	
370	---	
371	#1 is too long. Truncated to '#2'	
372	---	
373	---	
374	---	
375	---	
376	---	
377	Occurrence field not found in record	
378	`Depending-on´ field invalid	
379	---	
380	---	
381	---	
382	---	
383	---	
384	'Auto reports' not allowed in Target File	In case the transfer command in a record input procedure is used, the option" auto reports" is not allowed in the Target File.
385	Invalid in sorted Report Initial Procedure	
386	---	

Number	Message	Explanation
387	EBCDIC/Multi-record conflict for	Multiple record files containing EBCDIC fields in more than one record definition are not supported by a MicroFocus COBOL compiler.
388	Record must have a Storage Key	This message appears when no storage key has been found for a certain action on a record for which a storage key is necessary.
389	EBCDIC field overlap conflict for	Overlapping EBCDIC fields within an EBCDIC Source File are not supported by a MicroFocus COBOL compiler. If overlapping fields are used in this case, this message is returned.
390	---	
391	Duplicate definitions found for #1	
392	File changed since it was last verifiied	
393	Storage key type invalid for	
394	Record not in use	
395	Cannot exclude subsequent record	
396	`Depending-on´ field may not overlap #1	
397	Invalid in Record Input Procedure	
398	Entry record must have a Storage Key	If you have a CONTROLLED BY IDMS Source File for which the entry record does not have a storage key defined, this message is returned. You should add a storage key to the entry record or choose another entry record with a storage key.
399	DB Record: Invalid storage key or DB name (obsolete message)	
400	Invalid record for DB link (obsolete message)	

Messages 401 to 450

Number	Message	Explanation
401	Multiple Members invalid in DB link (obsolete message)	
402	---	
403	Field name is the same as File or Record name	

Number	Message	Explanation
404	Invalid path for DB File (obsolete message)	
405	Verifying Datacom file name	
406	Execution mode IMS/ Restartable not supported on IBM/ODBC	
407	Datacom file needs storage key index	
408	No TOTAL lines defined for printing/putting	
409	Command invalid in Group procedure	
410	Invalid random file key format	
411	---	
412	---	
413	CONVERT must have exactly 1 detail	
414	---	
415	Not allowed for CONVERT command	
416	TIME must be a numeric field	
417	TIMESTAMP holder type should be CHARACTER or NUMERIC	If a field is defined with a timestamp notion, but the data type is incorrect, this message is returned.
418	Date format must be 'YYYYMMDD	Sometimes only the ISO date format 'YYYYMMDD' is supported, e.g. for SQL Source files. In these cases a pre-compiler option can be set to read every date in the ISO date format.
419	Zoned data changed to SQL decimal	
420	Warning on Source File	A warning was issued on a source file. This error is used by MetaSuite to provide the file name on which the warning occurred. The kind of warning can be seen on the next line.
421	EXEC-SQL not supported for ODBC on IBM	
422	---	
423	Error on Target File	A error was issued on a target file. This error is used by MetaSuite to provide the file name on which the error occurred. The kind of error can be seen on the next line.
424	Hogan is no longer supported	From version 7.2 onwards, Hogan is no longer supported.
425	Fujitsu compilers do not support:	Some options are supported only by a limited number of compilers. For instance, EBCDIC mode on index sequential files is not supported by Fujitsu compilers.

Number	Message	Explanation
426	SYS-RAW not available after SORT	Numeric fields are stored in another format when they are copied to a sort file. This is for performance reasons. After the SORT the function SYS-RAW can not be used on them. The result would be misleading.
427	SQL used in Program Initial Procedure	This message only occurs on UNIX and Windows systems. Certain SQL dialects and some ODBC databases do not support some of the SQL statements used just after program load. If the *MOVE-PROGINIT-IF-SQL* option is set to *ON*, the program initial procedure will be executed after opening the files, and after the File Initial procedure.
428	Field #1 must be defined as a nullable field	When using the keywords SYS-STATUS and STATUS-INCLUDED in embedded SQL, the field must be defined as a nullable field.
429	Multiple occurring levels	Multiple occurring levels are not supported for target fields and work fields. *Multiple occurring* means: occurring field within a group field that is also occurring. (This results in multi-dimensional tables.)
430	Comparing dates with non-dates	All fields of type *date* are converted into numeric fields of type *YYYYMMDD*. The advantage of this method is that it is possible to compare character dates with numeric dates. The disadvantage is that you cannot compare dates of type character with character fields. You can bypass this error by using redefinition.
431	The usage of SYS-EDIT on a date field is superfluous	All fields of type *date* are converted into numeric fields of type *YYYYMMDD*. Therefore the usage of SYS-EDIT on date fields is not possible.
432	Invalid build number - must be 2 digits long	Change Default Build: The dictionary build number is a number between 00 and 99.
433	SQL data type of #1 not supported	
434	SQL data type of #1 has an invalid size	
435	---	
436	FORMAT number expected	
437	FORMAT number invalid	
438	Only allowed for accumulable numeric field	
439	Only allowed in Printable Detail Line	
440	---	
441	Cannot convert NUMERIC fields larger than 8 numbers	
442	FULL requires VARCHAR field	
443	---	

Number	Message	Explanation
444	Invalid century - must be 2 digits	
445	No User tables defined	
446	Renaming may affect existing programs	
447	---	
448	---	
449	---	
450	Invalid Break Year specified	

Messages 451 to 500

Number	Message	Explanation
451	POSITION required for in-stream field	
452	Table not found in SQL Catalog	
453	---	
454	---	
455	---	
456	---	
457	---	
458	---	
459	---	
460	---	
461	Error occurred in module #1	
462	---	
463	In-stream file command out of sequence	
464	Creator name exceeds the limit of 8 characters	
465	Creator required for SQL record	
466	`Of´ not allowed for SQL column	
467	Table name #1 has been put before #2	
468	Table name expected	

Number	Message	Explanation
469	WHERE clause required - not found	An SQL Source File which is controlled needs to have a WHERE clause defined in the path. If the WHERE clause is not present, this message is returned.
470	Maximum of 12 tables exceeded	
471	---	
472	---	
473	Column name expected	
474	---	
475	---	
476	---	
477	---	
478	---	
479	---	
480	---	
481	---	
482	---	
483	Key invalid for SQL file	
484	---	
485	SORT not allowed for EXEC=IMS	If a program is generated with a Execution default IMS, a Sort can not be done on non-SQL Source Files or on Target Files. If the Sort is specified anyway, this message is returned.
486	---	
487	Nested OCCURS not allowed for in-stream fields	
488	---	
489	SYS-SQL-NULL field not nullable	
490	SYS-SQL-LENGTH field not variable	
491	---	
492	---	
493	---	
494	---	
495	---	
496	---	
497	---	
498	---	

Number	Message	Explanation
499	Schema name expected	On an IDMS file definition, a schema name must be added. If the schema name is not what MetaSuite Generator expects it to be, this message is returned.

Messages 501 to 550

Number	Message	Explanation
501	---	
502	Option not allowed for this file	
503	---	
504	---	
505	---	
506	Uable to load table	
507	---	
508	Record/Block size invalid for COBOL	If the maximum record or block size for a file, or the record size for a record, is too big, this message is returned.
509	---	
510	No GET command for controlled file:	
511	---	
512	Keyword SQL expected	
513	Keyword, host variable or name expected	
514	INTO clause expected	
515	---	
516	Host variable expected	
517	SQL statement too long	
518	Field list or INTO clause expected	
519	SQL syntax error detected by DBMS	
520	SQL syntax warning detected by DBMS	
521	Maximum number of 'statement' host variables exceeded	
522	Maximum number of 'program' host variables exceeded	
523	---	

Number	Message	Explanation
524	---	
525	---	
526	SQL DDL statement not supported	
527	Keyword DIALECT expected	
528	Invalid SQL dialect specified	
529	---	
530	---	
531	---	
532	---	
533	---	
534	Only one set dialect allowed	
535	Cursor name expected	
536	---	
537	Cursor name expected	
538	Cursor name invalid	
539	Cursor not declared	
540	Maximum number of cursors exceeded	
541	Keyword OF expected	
542	Keyword END-EXEC expected	
543	Invalid within shared procedures	
544	---	
545	---	
546	The VALUE clause is missing for the 88-field #1	
547	Keyword NOT may only precede 88-level name	
548	---	
549	---	
550	Only one title allowed for Oracle	

Messages 551 to 600

Number	Message	Explanation
551	---	
552	---	
553	Invalid system limits table	
554	Call Technical Support	
555	---	
556	---	
557	Summary must have a TOTAL line	
558	PREPARE must be ON or OFF	
559	---	
560	Only one heading required (Sybase)	
561	'NOAUTO heading' option is not allowed (obsolete message)	
562	SQL dialect not set	If you want to generate MGL for a program that has an SQL source file, and no SQL dialect is set in MetaSuite Generator, this message will be returned.
563	IMPLEMENT instruction: table overflow	
564	User-defined edit mask not used for:	This message is displayed when specifying a superfluous edit-mask for a field. This edit-mask will not be used, so the generator gives this warning in order to notify the user.
565	Variable not defined	This is an MTL message. The MTL variable used in this expression is not yet defined by the SET instruction.
566	Maximum number of variables reached	This is an MTL message. The number of MTL variables exceeded the limit. For more information, please refer to the chapter MetaSuite Template Language (page 96).
567	Protected variable cannot be modified	This is an MTL message. A protected variable cannot be modified by the SET instruction.
568	IMPLEMENT instruction not allowed in MRL table	
569	FOR statement syntax error	This is an MTL message. For more information about the FOR statement, please refer to the chapter MetaSuite Template Language (page 96).
570	General MTL message	This is an MTL message, issued by the MSG, WARNING or ERROR command. For more information, refer to the chapter MetaSuite Template Language (page 96).

Number	Message	Explanation
571	Label is missing	This is an MTL message. The label specified in a "GOTO" or "SKIP" instruction has not been found. For more information about the jump instructions, refer to the chapter MetaSuite Template Language (page 96).
572	THEN/ELSE/END-IF without IF	This is an MTL message. The instruction "THEN" or "ELSE" or "END-IF" has been found, without corresponding "IF" instruction belonging to the same level. Possible reasons: a jump statement within an IF structure, wrong mixing of different structures, simply forgetting to code the IF instruction.
573	Label longer than 8 characters	This is an MTL message. A label name may not be longer than eight characters.
574	END-EVALUATE or END-IF missing	This is an MTL message. An IF-structure has to be closed by the END-IF instruction. An EVALUATE-structure has to be closed by the END-EVALUATE instruction. If this is not the case, this message will be displayed. For more information about IF and EVALUATE structures, please refer to the chapter MetaSuite Template Language (page 96).
575	Corresponding EVALUATE is missing	This is an MTL message. The instruction "WHEN", "THRU" or "END-EVALUATE" has been found, without corresponding "EVALUATE" instruction belonging to the same level. Possible reasons: a jump statement within an EVALUATE structure, wrong mixing of different structures, simply forgetting to code the EVALUATE instruction.
576	EVALUATE/IF - Stack full	This is an MTL message. The number of nestings exceeds the limit of 16. For more information about IF and EVALUATE structures, please refer to the chapter MetaSuite Template Language (page 96).
577	Syntax error in MTL statement	General MTL syntax error. Please refer to the chapter MetaSuite Template Language (page 96).
578	Numeric field compared with non-numeric field	In IF-structures numeric and non-numeric expressions cannot be compared to each other. Please refer to the chapter MetaSuite Template Language (page 96) for more information concerning the IF, AND and OR instructions.
579	---	
580	Division by zero	This is an MTL message. SET instruction: division by zero is not allowed. Please refer to the chapter MetaSuite Template Language (page 96) for more information concerning the SET instruction.
581	Commandline overflow	This is an MTL message. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
582	FOR/NEXT - Stack full	This is an MTL message. The number of nestings exceeds the limit of 16. Please refer to the chapter MetaSuite Template Language (page 96) for more information.

Number	Message	Explanation
583	STEP value not numeric	This is an MTL message. The STEP-expression in the FOR-loop must have a numeric result. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
584	TO value not numeric	This is an MTL message. The TO-expression in the FOR-loop must have a numeric result. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
585	FOR value not numeric	This is an MTL message. The FOR-expression in the FOR-loop must have a numeric result. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
586	---	
587	NEXT without FOR	This is an MTL message. The instruction "NEXT" has been found, without corresponding "FOR" instruction belonging to the same level. Possible reasons: a jump statement within a FOR structure, wrong mixing of different structures, simply forgetting to code the FOR instruction, or putting it in REMARK Please refer to the chapter MetaSuite Template Language (page 96) for more information.
588	FOR without NEXT	This is an MTL message. The instruction "FOR" has been found, without corresponding "NEXT" instruction belonging to the same level. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
589	STEP value is zero	This is an MTL message. The STEP-expression in the FOR-loop must have a non-zero result. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
590	Characters not allowed here	This is an MTL message. Characters may not be part of a numeric expression. If you want to work with variables in a numeric expression, please enclose them by straight brackets. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
591	OR/AND without IF	This is an MTL message. The instructions "OR" or "AND" have been found, without corresponding "IF" instruction belonging to the same level on a previous line. Please refer to the chapter MetaSuite Template Language (page 96) for more information.
592	THRU command without WHEN	This is an MTL message. The instruction "THRU" has been found, without corresponding "WHEN" instruction on the previous line. Please refer to the chapter MetaSuite Template Language (page 96) for more information.

Number	Message	Explanation
593	Wrong use of the reserved word OTHER	This is an MTL message. The reserved word *OTHER* has been used outside an *EVALUATE* instruction block, or on a wrong place. (By example in a THRU expression) Please refer to the chapter MetaSuite Template Language (page 96) for more information.
594	Invalid MTL function	
595	Syntax error in Function command	
596	String function - invalid position	
597	CALL level higher than 16	
598	Referenced dictionary table missing or invalid	
599	Invalid FREEZE/UNFREEZE option	
600	Warning! License key expires	Please contact your MetaSuiteadministrator or IKAN Solutions N.V. in order to receive and install a new license key.

Messages 601 to 650

Number	Message	Explanation
601	Null value will be ignored	First Case: Some field types like BINARY and FLOATING-POINT do not support the use of internal NULL indicators. The NULL setting will be ignored. Second Case: Null values cannot be used on target files of type CONTROLLED SEQUENTIAL, because of confusing redefinition possibility.
602	Occuring Null fields not allowed	This feature is currently not supported. A field with DBNAME INNULL, OUTNULL or OUTNULR cannot be an occurring field.
603	Notnull field has no Null indicator	This warning is displayed when assigning a SYS-NULL-VALUE to the SYS-STATUS of a field that contains no NULL-indicator. In other words: the DBNAME setting and the SYS-NULL setting are in conflict with each other. The SYS-STATUS of a field can be set to SYS-NULL-VALUE, even if the field contains no NULL-indicator (DBNAME is NOTNULL). The field will be initialized by this action!
604	Invalid hexadecimal literal	
605	Sort field of invalid type	This error occurs if a sort field has field type FLOAT or BIT. Those field types can give incorrect sort results due to the limitations of the sort process.

Number	Message	Explanation
606	SQL dialect set to ODBC	This message is returned if the SET SQL DIALECT is issued for a non-ODBC dialect, while the COBOL environment is FUJITSU. The SET SQL DIALECT command is overruled.
607	Invalid field type specified	
608	Array key not equal to Sort key	It is obvious that an external array containing a search key may not be sorted using another key. If done so, the SEARCH instruction will give unpredictable results.
609	Sort key in tree should be ascending	For an external array with a search key in MetaSuite, the search key is generated in ascending order. If you perform an initial sort to the array, the order of the first key (this must be the same key as the search key) has to be ascending.
610	Occurring fields not allowed	If CODE-CONTROL is set to a delimited record type, then no occurring fields may be included in the record description.
611	Redefined fields not allowed	If CODE-CONTROL is set to a delimited record type, then no redefining fields may be included in the record description.
612	No SORT or ORDER BY on array key	No SORT or ORDER BY is found when reading the external array file into a binary tree. If the read sequence is not correct, the external array will not function as it should. No SORT or SQL ORDER BY is found either. If the user is certain that the read order will be correct, this warning is superfluous.
613	Standard code-control table overruled	This message appears if the CODE-CONTROL parameter is specified for a file type that already uses a special type specific code-control table. It is just a warning. Special types are for instance DELIMITED and XML.
614	Record name expected	The Target File/DETAIL sentence contains a RECORD parameter in which the name of the record can be specified. If the name is missing, this message will be shown.
615	Column not found in path statement	The column (prefix) referred here is not specified in the preceding path statement.
616	---	
617	Multi-record Input Procedure not supported	This error is returned if you define an input procedure on a record that may occur more than once in a path.
618	Code-control table not defined	The code-control table, specified in the ADD FILE command, can't be found in the dictionary. The default code-control table will be taken. For more information about code-control tables, please refer to the section Code-control Tables (page 64).
619	SQL dialect should be ODBC	ODBC is the only SQL dialect supported by MetaSuite for this compiler.
620	Invalid quote setting	The correct syntax is: CHANGE DEFAULT QUOTE TO value, where value can be SINGLE, DOUBLE or ORIGINAL.
621	Usage of SYS-STATUS not allowed on #1	Function fields do not contain a status field. The usage of SYS-STATUS is therefore not permitted on function fields.

Number	Message	Explanation
622	Function field: reset to NOTNULL	Function fields do not contain a status field. For that reason the "INNULL", "OUTNULL", or "OUTNULR" definition is automatically reset to "NOTNULL".
623	---	
624	Invalid decimal point setting	CHANGE DECIMAL POINT TO COMMA or POINT. Any other value will result in this error.
625	Field size does not match	The field size does not match the edit or date mask.
626	Multiple records not possible for	For certain file types like delimited files, it is not possible to define multi-record structures, because you have to know which record type you are parsing before parsing it. But in the case of multi-record structures, you only know the record type after having read the record.
627	Invalid index (zero or too high)	Invalid reference towards a file, record or field MTL-variable.
628	This prefix is reserved	Prefixes *FILE-*, *RECORD-* and *FIELD-* are reserved. Define MTL variables with other prefixes.
629	Invalid special variable name	Prefixes *FILE-*, *RECORD-* and *FIELD-* are reserved. The MTL variable that you are referring to is not supported.
630	Invalid SQL action	SQL action can be INSERT, UPDATE or DELETE.
631	IDMS syntax error: Expected	Invalid expression. The IDMS syntax checker expects the (or one of the) specified keyword(s).
632	Colon not permitted in table name	A colon (:) should not appear in a table name. MetaSuite does not support this feature yet.
633	Edit mask required for PRN fields	For each PRN field, an edit mask has to be defined.
634	No Control Key correspondence found	The Control Key is not found.
635	Warning! Forked path used in	The controlling file key field is a work field. A test on "forked paths" is useless in this case. The user is responsible for proper setting of the field in the controlling file input procedure.
636	No space reserved for OUTNULR byte	This field is defined as "OUTNULR". This means that on the right side of the field, one byte should be reserved for the null character. This space is not available.
637	No space reserved for OUTNULL byte	This field is defined as "OUTNULL". This means that on the left side of the field, one byte should be reserved for the null character. This space is not available.
638	Subpath depends on occurring path	This type of structures is not supported yet.
639	Invalid Target Field or Work Field	A target field or work field cannot be a field of type 'BIT'. This is not supported yet.
640	Expression can not be interpreted:	MTL expression cannot be validated because different types of MTL variables cannot be compared to each other. Numeric and non-numeric data cannot be compared to each other.

Number	Message	Explanation
641	Parenthesis not closed yet	For instance: the parenthesis of the ORDER clause should be closed before starting the WHERE clause.
642	Parenthesis closed too early	The parenthesis for the SELECT/WHERE clause closed too early.
643	Field #1 should be alphanumeric	Functions like SYS-TRIM and SYS-UPPERCASE can only be used with alphanumeric fields, not with numeric fields.
644	Double accumulation might occur	When combining detail lines and total lines in one report, the accumulation logic might be performed twice. Be careful in this case.
645	DEPENDING ON not allowed. Reason:	Tables with a variable number of elements are not allowed in this case. In case of function files, for instance, it is not allowed to have multiple occurring fields with a varying occurrence.
646	Nested #1 structure not allowed	This feature is not yet supported.
647	Row terminator part of delimiter	The user has specified a delimited file containing column separators (or delimiters) and a row terminator. In order to avoid confusion, the row terminator must not be a part of the column separator.
648	No fixed variant of code table	The output control table #1 is used for variable target files. The corresponding output control table for fixed target files has not been defined.
649	Array key #1 does not belong to #2	The array key #1 should be a part of the array file #2
650	Prefix not supported for XPC files	XPC = XML with Path Control. In the current version, XML files with path control are supported for source files without the file prefix.

Messages 651 to 700

Number	Message	Explanation
651	Key value length not equal to key length	The external array key and the key value to which it is compared, have not the same size.
652	Maximum number of key values is #1	
653	Numeric date mask required for #1	
654	GET command: more keys than values	
655	Index field #1 is too small	Multiple occurring field has an index field which is too small for the number of fields that it can contain.
656	Function #1 not supported in IF/CASE	This functionality is not supported in an IF-block or within a CASE-block.

Number	Message	Explanation
657	#1 weakly supported in IF/CASE	This functionality is not always ported in an IF-block or a CASE-block. Please contact MetaSuite Support.
658	Maximum number of total values is #1	
659	Invalid argument value	Several possibilities: 1. (Formatted XML Module) The XML-NAME keyword is used, but no XML name was provided. 2. (Formatted XML Module) The XML-TYPE keyword is used, but no XML type is provided or it is invalid (valid values are "ATTRIBUTE" OR "GROUP"). 3. The second parameter of SUBSTRING function is invalid or omitted. 4. The third parameter of SUBSTRING function is invalid or omitted.
660	Invalid or no format mask	Several possibilities: 1. (Formatted XML Module) A keyword (XML-PATH, FORMAT-MASK, XML-DECLARATION) with a long property value (250 characters) has no value. 2. (Unicode Module) An entry in the dictionary table UCD001 is invalid (CCSID in five positions followed by '-', 'E', 'A' or 'U').
661	---	
662	Compute statement overflow (> 20 lines)	
663	Maximum file number reached	The MetaMap model contains more than 99 source files.
664	Maximum record number reached	The MetaMap model contains more than 99 source records.
665	Invalid CCSID (File, record, field)	Several possibilities: 1. (Unicode Module) The MetaMap model contains an invalid CCSID (less than zero or greater than 65535 - X'FFFF') at file, record or field level. 2. (Unicode Module) The MetaMap model contains the CCSID keyword, but no CCSID was provided. 3. (Unicode Module) The MetaMap model contains the CCSID keyword, but the CCSID provided with this keyword is not listed in the dictionary table UCD001.
666	#1: #2 CCSID #3 invalid for type #4	Several possibilities: 1. (Unicode Module) An invalid CCSID (not a Unicode CCSID) is provided for a Unicode data field. 2. (Unicode Module) More than 3.650 CCSIDs need to be stored in table ESYM-2B-ADDED. 3. (Unicode Module) All CCSIDs used in a MetaMap model need to be declared in the UCD001 dictionary table.

Number	Message	Explanation
667	#1: CCSID #2 not valid for type #3	The UCD001 dictionary table declares the valid CCSIDs and their type (ASCII, EBCDIC, UNICODE). All CCSIDs mentioned in a MetaMap model need to be used according those rules. For example: CORRECT = a Unicode CCSID for a Unicode data field INCORRECT = an ASCII CCSID for a Unicode data field (Unicode Module).
668	Function #1 not allowed	A function needs to have a name (MetaSuite or user defined). The valid MetaSuite functions are: ABSOLUTE-VAL, ASCII, ASCII-UNICODE, ASCII-UTF8, BINARY, CURRENT-DATE, EBCDIC, EBCDIC-UNICODE, EBCDIC-UTF8, HEXADECIMAL, INTEGER-OF-DATE, LOWER-CASE, NUMVAL, REVERSE, SYS-LENGTH-R, TRIM, UNICODE-ASCII, UNICODE-EBCDIC, UPPER-CASE, UTF8-ASCII, UTF8-, EBCDIC, WHEN-COMPILED
669	Invalid column separator (1-3 chars)	The COLUMN-SEPARATOR keyword of a delimited file is invalid (valid example: COLUMN-SEPARATOR ';').
670	Invalid row terminator (1-3 chars)	The ROW-TERMINATOR keyword of a delimited file is invalid (valid example: ROW-TERMINATOR '\\')
671	Invalid Date. The date format has to be YYYYMMDD	When assigning a date value to SYS-DATE, the User should take care that the format in which this date value is specified is YYYYMMDD.
672	#1-Missing #2 parenthesis	The syntax of a function mentions the sequence of the mandatory and optional parameters. The parameters need to be put between left and right parenthesis. This message can appear when the number of parameters and their separating commas is wrong. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example “INSTRING” #2 is the position of the parenthesis, for example “Opening” As in “INSTRING-Missing Opening Parenthesis”
673	#1-Wrong parameter #2	The syntax of a function mentions the sequence of the mandatory and optional parameters. The parameters need to be put between left and right parenthesis. This message can appear when the number of parameters and their separating commas is wrong. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example “INSTRING” #2 is the position of the parameter, for example “3” As in “INSTRING-Wrong Parameter 3”

Number	Message	Explanation
674	#1-Missing parameter #2	The syntax of a function mentions the sequence of the mandatory and optional parameters. The parameters need to be put between left and right parenthesis. This message can appear when the number of parameters and their separating commas is wrong. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example "INSTRING" #2 is the position of the parameter, for example "2" As in "INSTRING-Missing Parameter 2"
675	#1-Wrong separator #2	The syntax of a function mentions the sequence of the mandatory and optional parameters. The parameters need to be put between left and right parenthesis. This message can appear when the number of parameters and their separating commas is wrong. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example "INSTRING" #2 is the position of the separator, for example "3" As in "INSTRING-Wrong Separator 1"
676	---	
677	#1-Parameter #2 must be #3	The syntax of a function mentions the sequence of the mandatory and optional parameters. The parameters need to have a specific data type. This message can appear when the number of parameters and their separating commas is wrong. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example "INSTRING" #2 is the position of the parameter, for example "1" #3 is the data type the parameter should have, for example "Alphanumeric" As in "INSTRING-Parameter 1 Must be Alphanumeric"
678	---	
679	#1-Strings encoding different	String functions work on ASCII, EBCDIC and Unicode strings, but all string parameters must be of the same type. #1 is the function name, for example "INSTRING" As in "INSTRING-Strings Encoding Different"
680	#1-Too many parameters	The syntax of a function mentions the number of mandatory and optional parameters. This number may not be exceeded. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example "INSTRING" As in "INSTRING-Too Many Parameters"
681	#1-Parameter beyond closing parenthesis	The syntax of a function mentions the number of mandatory and optional parameters. The end of the function call is the closing parenthesis, no parameters are allowed beyond this point. Compare the syntax of your function call in MetaMap against the syntax diagram provided by IKAN. #1 is the function name, for example "INSTRING" As in "INSTRING-Param Beyond Closing Parenthesis"

Number	Message	Explanation
682	Line Sequential file contains a signed field	Float, binary, packed and zoned data types are not allowed with line sequential files.
683	Line Sequential file contains a Unicode field	National and PRN-National data types are not allowed with line sequential files.
684	Usage of Unicode only supported for IBM	Unicode data types are only supported for the IBM target environments (Unicode Module).
685	Edit mask length different from size	For PRN fields: the edit mask should be equal to the field size. CORRECT = TYPE PRN SIZE 9 DECIMAL 2 EDIT 'Z(6)9V9(2)' total of the mask = 9 (6+1+2) INCORRECT = TYPE PRN SIZE 9 DECIMAL 2 EDIT 'Z(7)9V9(2)' total of the mask = 10 (7+1+2)
686	#1 is an invalid CCSID (not declared)	A CCSID can only be assigned to an A (ASCII), E (EBCDIC), or U (UNICODE) field. Their could be an error in the UCD001 dictionary table.
687	Mask contains decimals; no decimals have been defined	The edit mask contains decimals (as in 'Z(7)9V9(2)') but the data field was declared as having no decimals.
688	Decimals not the same as in Edit mask	The edit mask contains a number of decimals (as in 'Z(7)9V9(2)'), but the data field was declared with a different number of decimals.
689	Size of 'replaced' and 'replacement' strings not equal	The length of the string to be replaced has to be equal to the length of the replacing string. #1 is the function name, for example “REPLACE-ALL” #2 is the position of the parameter, for example “Different” As in “REPLACE-ALL – Lengths ToBeRepl & Replacing Different”
690	---	The length of the string to be replaced has to be equal to the length of the replacing string. #1 is the function name, for example “REPLACE-ALL” #2 is the position of the parameter, for example “Length” As in “REPLACE-ALL – ToBeReplaced Length > String Length”
691	Model without any Target File	A MetaMap model should contain at least one target file.

692	Unknown #1-function #2	The OBFCVT table declares all the User-defined Functions. It makes the link between the external name as used in MetaMap and the internal name within the MetaSuite generator. All User-defined Functions, the IKAN Certified Functions included, have to be listed in the OBFCVT table. The external name should be used within MetaMap (Obfuscation Module).
		OBFCVT
		`>* Function Name-----> OBx R M Parameters`
		`RANDOM-INIT 001 N 0 N`
		MetaMap
		MY-TARGET-FIELD = RANDOM-INIT (123456789).

Number	Message	Explanation

693	Function number invalid	The OBFCVT table declares all the User-defined Functions. It creates the link between the external name as used in MetaMap and the internal name within the MetaSuite generator.
		The table name has two parts:
		1. the three mandatory characters "OBP"
		2. a number between 001 included and 999 included that has to be specified in the OBx column of the OBFCVT table.
		Example:
		`>* Function Name-----> OBx R M Parameters`
		`RANDOM-INIT 001 N 0 N`

694	---	
695	Logic for #1-FUNCTION #2 not found	The OBFCVT table declares all the User-defined Functions. It makes the link between the external name as used in MetaMap and the internal name within the MetaSuite generator. All User-defined Functions, the IKAN Certified Functions included, have to be listed in the OBFCVT table. The function's data and logic are implemented as a dictionary table (OBFnnn) that must exist in the MetaSuite Dictionary. This table was not found (Obfuscation Module).
696	Not enough mandatory parameters	The OBFCVT table mentions, for each function, how many mandatory parameters must be provided at function call time. This message appears if the number of parameters provided in MetaMap is lower than the number declared in the OBFCVT table (Obfuscation Module).

697	Too many parameters provided	The OBFCVT table mentions, for each function, how many mandatory parameters must be provided at function call time and the data type of each parameter, including the optional ones. You have to provide the correct number of parameters as declared in the OBFCVT table (Obfuscation Module).
		OBFCVT
		`>* Function Name-----> OBx R M Parameters`
		`RANDOM-INIT 001 N 0 N`
		`only one parameter`
		MetaMap
		MY-TARGET-FIELD = RANDOM-INIT (123456789, "this parameter is not valid !").

698	Too many parameter literals in model	The maximum of 999 literals in a MetaMap model was reached (Obfuscation Module).
699	#1 used as System and User Function	
700	No absolute XPath available for XML	This message is applicable for XML files. Record path is a relative path, but the file path, to which it is relative, is not available.

Messages 701 to 711

Number	Message	Explanation
701	File Path should be an absolute path	
702	XPath not available for:	XPATH for the record should be specified on record level, if the XPATH on file level is missing.
703	Attribute on File Path: not supported	No XPATH specified on record level. Field on the highest level is an attribute. It should be attached to the record, for which no XPATH was specified. Solution: make the XPATH of the file one level shorter, and add the level that was omitted to the XPATH of the record(s).
704	Field #1 does not belong to an external array	The usage of SYS-CURRENT-KEY and SYS-RANDOM-KEY within an array is limited to external arrays.
705	#1 not allowed in File Initial Procedure	The usage of this keyword is not allowed in File Inital Proc. For instance: SYS-RANDOM-KEY is not allowed in File Inital Proc
706	Empty node in XPATH of Target Record #1	The XPATH of the target record is not empty, but some nodes in it are. For instance: when the XPATH is '/X/Y//Z', there is a gap between Y and Z. These kind of XPATHS are invalid.
707	String expression too long	The string expression exceeds the allowable limit. For instance: the string in a DEBUG statement can not exceed the size of 42 characters
708	Keyword HEADER not supported in this context	
709	END-CASE without CASE	The END-CASE statement was found without the corresponding CASE statement.
710	END-FOR without FOR	The END-FOR statement was found without the corresponding FOR statement.
711	FOR/END-FOR sequence invalid	The END-FOR statement has been put in an invalid place.

Index

A

ADD 33
ADD TABLE 91

B

Batch
    Calling the Generator 144
BUILD 25

C

CALL Table 104
Calling the Generator in Batch 144
    Example 145
    Preparing 144
    Using MSBGEN.EXE 144
CHANGE DEFAULT BUILD 86
CHANGE DEFAULT CHARACTER-CCSID 91
CHANGE DEFAULT DATE 86
CHANGE DEFAULT DECIMAL 87
CHANGE DEFAULT DYNAMIC 91
CHANGE DEFAULT EXEC 88
CHANGE DEFAULT INTERPUNCTION 88
CHANGE DEFAULT NULL 89
CHANGE DEFAULT NULLABLE 89
CHANGE DEFAULT PAGE 89
CHANGE DEFAULT QUOTE 90
CHANGE DEFAULT SQL 90
CHANGE DEFAULT SQL-QUOTE 30, 91
CHANGE DEFAULT UNICODE-CCSID 91
CHANGE Options 24
    BUILD 25
    CHANGE DEFAULT SQL-QUOTE 30
    CHARACTER-CCSID 32
    DATA FORMAT 25
    DATE BREAK YEAR 26
    DECIMAL 26
    DYNAMIC 31
    EXEC 27
    INTERPUNCTION 27
    LANGUAGE 31
    NULL 28
    NULLABLE 28
    PAGE 29
    QUOTE 29
    SQL 30
    UNICODE-CCSID 31
CHARACTER-CCSID 32
Code Control Tables 64
    Source Code Control 66
    Standard Table Format 65
    Target Code Control 68
    User-written Code Control Tables 71, 72
Command Wizard 21
    CHANGE Options 24
    TABLE Options 32
Commands
    Installation Language 85
    MTL 102
COPY 34
COPY TABLE 92
Create Dictionary/Enter License Key Screen 6
Creating a new Dictionary 6

D

DATE BREAK YEAR 26
DATE FORMAT 25
DECIMAL 26
Default Folders 82
DELETE 34
DELETE TABLE 93
Dictionary
    Creating 6
Dictionary Error Messages 151
Dictionary Options 12
Dictionary Options Screen 12
DUMP 105
DYNAMIC 31

E

Entering a License Key 10
EVALUATE Structures 105
EXEC 27
EXIT Table 107
EXPORT 35
Expressions
    MTL 98

F

Fault Message Handling 107
FOR-NEXT Loops 110
FREEZE 111
Functions
    MTL 99

G

Generate
    Implementing MIL Instructions 18
    Working with MDL Files 36
    Working with MSM Files 46
    Working with MXL Files 48
Generate Screen 15
Generator Initial Settings 81
Generator Manager Screens
    Create Dictionary/Enter License Key 6
    Dictionary Options 12
    Generate 15
    Other Functions 81
    Table Maintenance 53
Generator Message Reference 147
    Dictionary Error Messages 151
    Generator Messages 151
    Generator Return Codes 147
    Generator Syntax Error Messages 150
Generator Messages 151
Generator Return Codes 147
Generator Syntax Error Messages 150
Generator Variables 101
GOTO 111

I

IF Structures 113
Implementing MIL Instructions 18
    Adding MIL Instructions 21
    Command Wizard 21
    Copying Existing MIL Files 20
IMPORT 35
Initial Settings (Generator) 81
Installation Language
    ADD TABLE 91
    CHANGE DEFAULT BUILD 86
    CHANGE DEFAULT CHARACTER-CCSID 91
    CHANGE DEFAULT DATE 86
    CHANGE DEFAULT DECIMAL 87
    CHANGE DEFAULT DYNAMIC 91
    CHANGE DEFAULT EXEC 88
    CHANGE DEFAULT INTERPUNCTION 88
    CHANGE DEFAULT NULL 89
    CHANGE DEFAULT NULLABLE 89
    CHANGE DEFAULT PAGE 89
    CHANGE DEFAULT QUOTE 90
    CHANGE DEFAULT SQL 90
    CHANGE DEFAULT SQL-QUOTE 91
    CHANGE DEFAULT UNICODE-CCSID 91
    COPY TABLE 92
    DELETE TABLE 93
    LIST DEFAULT 93
    LIST TABLE 93
    LIST VERSION 94
    NEW 85
    REMARKS 95
Installation Language Commands 85
INTERPUNCTION 27

L

LANGUAGE 31
License Key 10
LIST 33
LIST DEFAULT 93
LIST TABLE 93
LIST VERSION 94
Logging on 3

M

MDL Files 36
    Delimited Target MDL 39
    Editing 38
    Generate MDL Screen 37
    Parse XML Source 44
    XML Target MDL 41
MetaSuite Installation Language 18
MGL Tables
    Code Control Table Facility 64
    Code Tables 71
    Customized COBOL Code Tables 74
    Customized Code Control Tables 73
    MGL Table Maintenance Screen 58
    Source Code Control 66
    Standard Code Control Table Format 65
    Target Code Control 68
    User-wrtitten Code Control Tables 71, 72
MIL Instructions 18
MRL Tables
    MRL Table Maintenance Screen 76
MSBGEN 144
MSM Files 46
MTL Commands 102
    CALL Table 104
    DUMP 105
    EVALUATE Structures 105
    EXIT Table 107
    Fault Message Handling 107
    FOR-NEXT Loops 110
    FREEZE 111
    GOTO 111
    IF Structures 113
    NESTED IF 114
    Notation Conventions 103

REMARK 114
SET 115
SKIP 115
TRACE 116
UNFREEZE 111
UNSET 117
MTL Expressions 98
MTL Field-related Variables 124
MTL Functions 99
MTL SourceFile-specific Variables 120
MTL SourceRecord-related Variables 121
MTL System Variables 118
MTL TargetFile-related Variables 122
MTL TargetRecord-related Variables 123
MTL Variables 97
MXL Files 48
Generating 49
Generating Multiple MXL Files in Batch 51

N

NESTED IF 114
NEW 85
NULL 28
NULLABLE 28

O

Other Functions
Browse Folders 82
Generator Initial Settings 81
Other Functions Screen 81

P

PAGE 29

Q

QUOTE 29

R

REMARK 114
REMARKS 95
Run Language 127

S

SET 115
SKIP 115
SQL 30
String Modules 134
System Variables
MTL 118

T

Table Maintenance
MGL Tables 57
Table Maintenance Screen 53
TABLE Options 32
ADD 33
COPY 34
DELETE 34
EXPORT 35
IMPORT 35
LIST 33
Template Language
Generator Variables 101
MTL Commands 102
MTL Expressions 98
MTL Field-related Variables 124
MTL Functions 99
MTL SourceFile-specific Variables 120
MTL SourceRecord-related Variables 121
MTL TargetFile-related Variables 122
MTL TargetRecord-related Variables 123
MTL Variables 97
Template Language Reference 96
TRACE 116

U

UNFREEZE 111
UNICODE-CCSID 31

V

Variables
Generator 101
MTL 97
Field-related 124
SourceFile-specific Variables 120
SourceRecord-related 121
System 118
TargetFile-related 122
TargetRecord-related 123